DSP TMS320C3x SERIES

PCI BUS

4-Channel High Speed Data Acquisition

DMP1162B

Software (*DmeXpress*) Manual

WEB : www.dmelec.com
TEL : +82-53-553-5500

< Assistance Contact >

DongMyung Elctronics Co, LTD.

Tel : +82-53-553-5500

Web : www.dmelec.com

Email : webmaster@dme.co.kr

< CAUTION >

CHANGES OR MODIFICATIONS NOT EXPRESSLY APPROVED BY THE PARTY RESPONSIBLE FOR COMPLIANCE COULD VOID THE USER'S AUTHORITY TO OPERATE THE EQUIPMENT

< NOTICE >

The information in this document is subject to change in order to improve reliability, design, or function without prior notice and does not represent a commitment on the part of this company.

In no event will we be liable for direct, indirect,special, incidental,or consequential damages arising out of the use or the inability to use the product or documentation, even if advised of the possibility of such damages.

< Trademark Acknowledgements >

< Limited Warranty >

Our company warrants this product against defects in materials and workmanship for a period of one year from the date of purchase. During the warranty period,a product determined by us to be defective in form or function will be repaired or at our option, to be replaced at no charge.

This warranty does not apply if the product has been damaged by accident, abuse, misuse, or as a result of service or modification other than by us.

This warranty is in lieu of any other warranty expressed or implied. In no event shall we held liable for incidental or consequential damages, such as lost revenue or lost business opportunities arising from the purchase of this product.

Table of Contents

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This manual covers software which runs on the DMP1162B and the application software which runs on the analysis computer(PC).
The software which runs on the DMP1162B is defined as "firmware" in this manual.
Miscellaneous software such as VxD, DLL, VHDL is described in this manual, which is used for DMP1162B operation.

The use of firmware is very simple.
After the firmware is downloaded to the DMP11622B, the firmware is going to be active.
The brief description and simple flowchart are provided in this manual, because the programming firmware is very sophisticated with relation to the DMP1162B hardware.
If you want to modify firmware, please contact to DongMyung Electronics.

The description of application software includes file handling, software configure setup, view setup, run (data acquisition), etc.
Application software can be developed under the Microsoft Visual C++ environment.
If you are experienced in Microsoft Visual C++ environment, modify source code or program new software referring to source code which has CD.
If you are not experienced in Microsoft Visual C++ environment, you have to study Microsoft Visual C++, at first.

This software manual provides software library whose function is used to firmware, application software of the DMP1162B.
It also provides reference documents which was referred at developing firmware, application software of the DMP1162B.

If you have any questions about firmware source code or application software source code of the DMP1162B, contact to DongMyung Electronics.

Chapter 2
Installation

This chapter describes how to install software of DMP1162B.
and contains verifying software of DMP1162B.
Term of the DMP1162B' software also are defined in this chapter.
Chapter contains

2.1 Environment

2.1.1 Run Time Environment

Table 2-1 Run Time Environment

Item	Description
Windows98	OS
VxD	Driver file
DmeXpress	Test SOFTWARE

2.1.2 Development Environment

Table 2-2 Development Environment

Item	Environment	Note
VHDL	Maxplus2	Board
Firmware	TMS320C3x Assembler, Linker, C Compiler	Software of Board
DLL	MSVC++ 6.0	PC
Application	MSVC++ 6.0	PC
VxD	MS Assembler, MSVC++6.0, SOFTICE, MS DDK	Board Driver of PC

2.2 Installing Software

Follow the procedures below to install software of the DMP1162B board.

1. Insert setup CD.

2. Follow the displayed window message.

3. Repeat upper item, until "complete setup" message.

2.3 Defining Term

Because the kinds of software mentioned in this manual are so complex, and the definition is ambiguous, term are defined at the following section.

2.3.1 VHDL

VHDL is defined as software which is used to programming of PLD chip (EPM7128).

For details, refer to "MAXPLUS2" software and related documents of "ALTRA" corporation.

2.3.2 Firmware

Firmware is define as software which runs on the DMP1162B Board.

(The definition of "firmware" is different from its original definition.)

It consists of ROM code and RAM code

2.3.2.1 ROM Code

It is EPROM program of the DMP1162B Board.

It consists of hardware initializing routine section and HEX code downloading routine section from PC.

2.3.2.2 RAM Code

It consists of program and data code which runs on RAM of the DMP1162B Board

The program will be processing after HEX code is downloaded from PC.

If you want to modify HEX code of the DMP1162B Board, modify firmware source code of the DMP1162B Board at appendix, and then do assemble processing, compile processing, link processing, hexcode processing.

RAM code consists of firmware library and application firmware.

1) Firmware library

Firmware library is functions that can be called from application firmware.

2) Application firmware

Application Firmware is a software that runs on the DMP1162B Board.

2.3.3 Software

Software mentioned in this manual is defined as running software in PC.

2.3.3.1 VxD

VxD is the Microsoft Window98 hardware driver file for the DMP1162B Board.

2.3.3.2 DLL

DLL is the Microsoft Window98 Dynamic Link Library to communicate VxD with application software(DmeXpress).

2.3.3.3 Application Software

1) Software Library

Software library is functions that can be called from application software.

2) Application Software

Application software is a software that runs on analysis computer (PC).

Application software is also defined as “DmeXpress”.

Chapter 3
Firmware Description

This chapter simply describes firmware of the DMP1162B.

Chapter contains

3.1 Description

Firmware consists of the following four parts.

1) Booting Part(File Down Loader)

- Booting device is EPROM(27C256).
- The contents of ROM is file down loader.
- About source code of ROM, contact to DongMyung Electronics.

2) Command Part(Sender, Receive)

- Mail Box0 is used to send command from Host to DSP.
- Mail Box2 is used to send command from DSP to Host.
- Command Spec(32Bit) is showed in the following table.

Table 3-1 Command Spec.

No.	Command	Hex Code	Note
1	A/D Begin	0x10	Auto Mode
2	A/D Begin	0x11	Trigger Mode
3	A/D Begin	0x12	Stationary Mode
4	A/D Stop	0x13	A/D Stop(Auto, Trigger, Stationary)
5	Trigger Level	0x30	Check Point
6	Trigger Channel	0x31	Trigger Mode, Trigger Channel setting
7	Pre-Trigger	0x32	Trigger pre10%, post90% (10% below)
8	Trigger Quantity Reference	0x33	Trigger (10%), Rectangular Window
9	Trigger Quantity Response	0x34	Trigger (10%), Rectangular Window
10	Gain Control	0x40	• 0 CH : 0x10 ~ 0x13 • 1 CH : 0x20 ~ 0x23 • 2 CH : 0x40 ~ 0x43 • 3 CH : 0x80 ~ 0x83 # Gain . X0 : 1, . X1 : 2, . X2 : 5, . X3 : 10
11	Timer Install	0x50	Frequency Install
12	Timer Reset	0x51	Frequency Reset
13	Timer Setup	0x52	Frequency Setup
14	Window for reference channel	0x60	0 : No Window 3 : Rectangular 1 : Hanning 4 : Hamming 2 : Exponential
15	Window for response channels	0x61	0 : No Window 3 : Rectangular 1 : Hanning 4 : Hamming 2 : Exponential
16	Exponential Decay Reference	0x62	Percent (%), Exponential Window
17	Exponential Decay Response	0x63	Percent (%), Exponential Window
18	Frame Size	0x70	512 ~ 8192 byte
19	Average Count	0x71	Stationary Variable
20	Overlap	0x72	Percent (%) - 0, 25, 50, 75 %
21	Acquisition Monitor	0x80	0: time 3: time & spectra 1: time & windowed 4: avg spectra 2: spectra

3) Data Tx, Rx Part(Sender, Receiver)

- Send and receive mass data from PC to DSP.
- 32Bit specification.
- For its use, open the apertures.

4) H/W Dignostic and Driver Part

- Memory : Read, Write Test
- Timer : On, Off Timer or Count.
- Amplifier : Amplifier Input signal.
- A/D : Converts Analog signal to Digital signal.

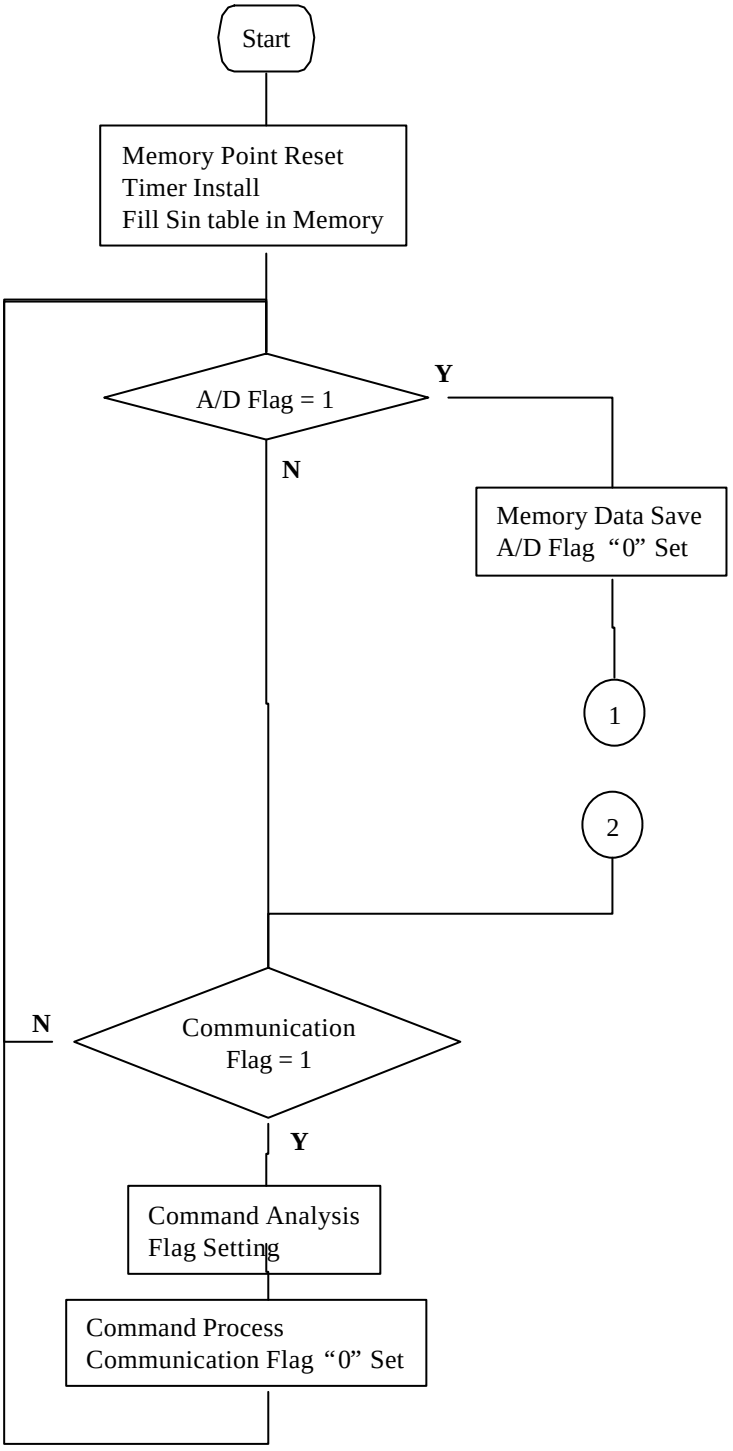
The DMP1162B' s firmware programming is very sophisticated.

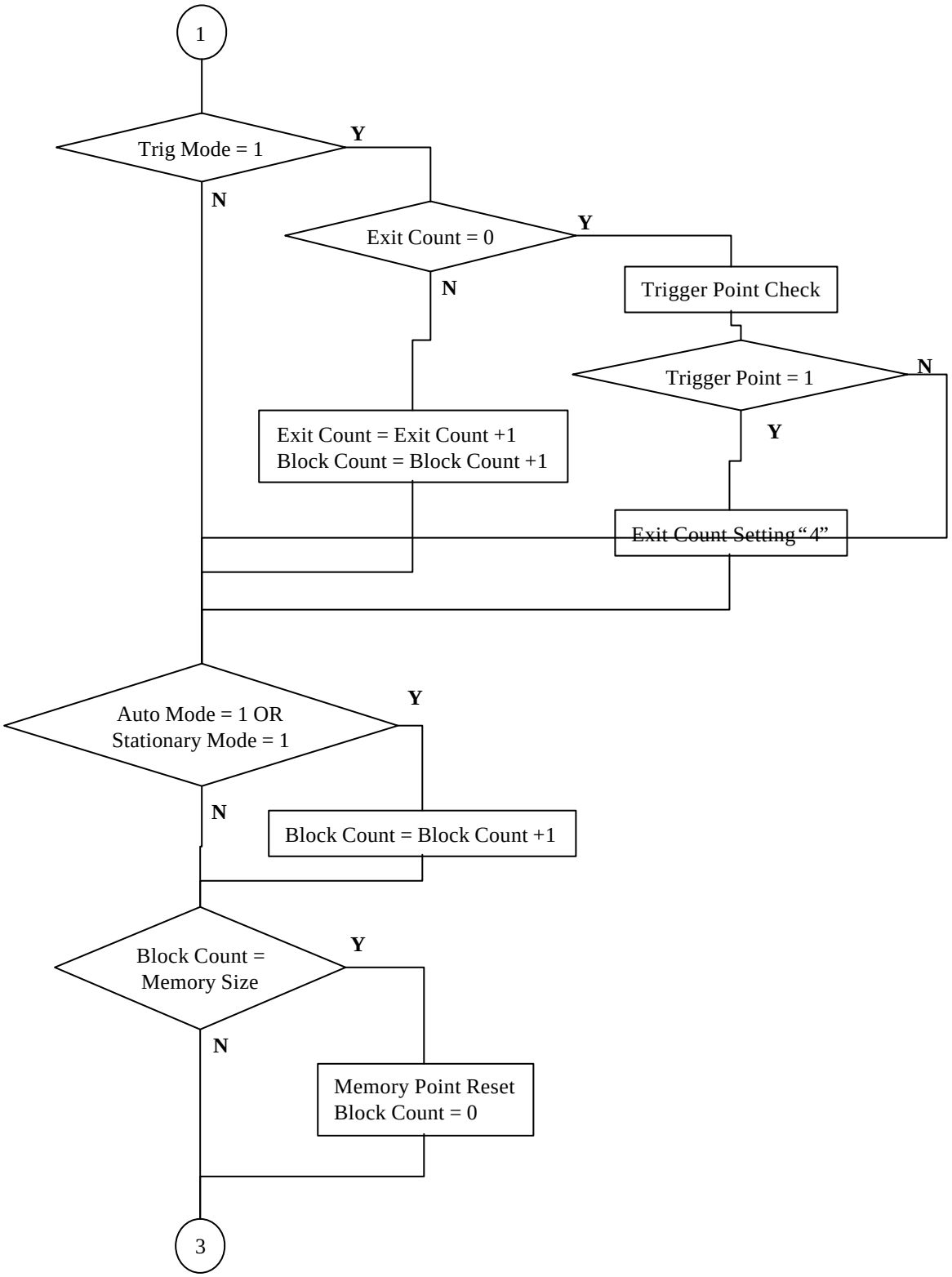
If you want to program, keep the following rules.

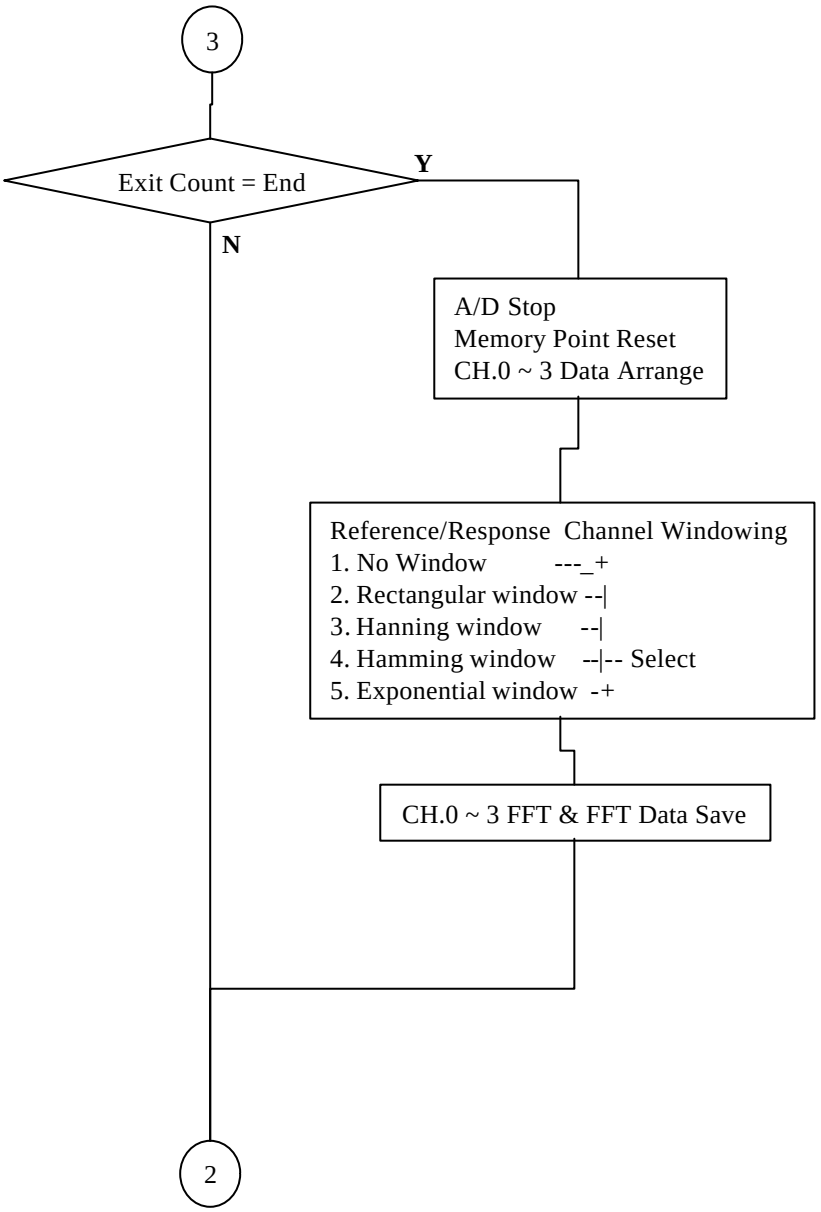
1. You have to install the development environment such as TMS320C3x assembler, linker and C complier.
   (Note: TMS 320Cx assembler, linker and C complier are not provided. If you want to develop firmware, you have to purchase the above tools from Texas Instrument,Inc.)
2. You should understand the DMP1162B hardware, TMS320C3x chip set, PCI chip set and ADC chip set. You should read DMP1162B hardware manual thoroughly. You also should read datasheets and application notes about TMS320C3x chip set, PCI chip set and ADC chip set.
3. After you met the above requirements, you can develop or modify the firmware using the source codes of appendix and library in the next chapter.

3.2 Flow Chart

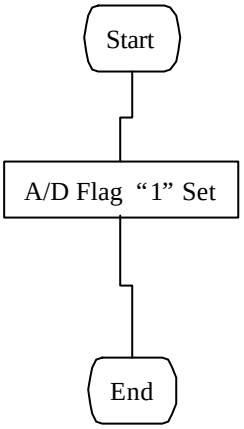
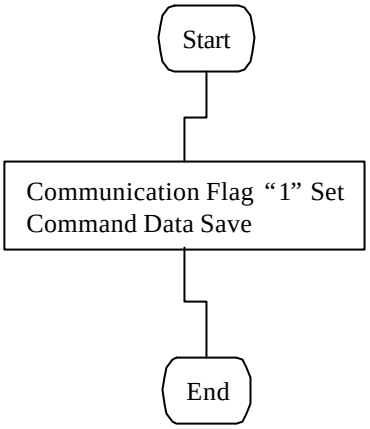
(1) Main Program & Subroutine




1
Trig Mode = 1
Y
N
Exit Count = 0
Y
N
Trigger Point Check
Trigger Point = 1
N
Y
Exit Count = Exit Count +1
Block Count = Block Count +1
Exit Count Setting "4"
Auto Mode = 1 OR
Stationary Mode = 1
Y
N
Block Count = Block Count +1
Block Count =
Memory Size
Y
N
Memory Point Reset
Block Count = 0
3


3
Exit Count = End
Y
N
A/D Stop
Memory Point Reset
CH.0 ~ 3 Data Arrange
Reference/Response Channel Windowing
1. No Window ---_+
2. Rectangular window --|
3. Hanning window --|
4. Hamming window --|-- Select
5. Exponential window -+
CH.0 ~ 3 FFT & FFT Data Save
2

(2) Interrupt Service Routine

A/D INT	COMMUNICATION INT
	

3.3 Function Block Description

Function block is described simply at source code of firmware.

The following table shows the file used at application firmware and brief contents.

The appendix also has the full source code of firmware.

Table 3-2 Firmware Filename

No.	Filename	Contents	Note
1	C31.c	Main program	
2	m31boot.asm	Interrupt & Communication Port setting	
3	Sub.asm	Subroutine	
4	Forfft.asm	FFT	
5	C31.cmd	Memory Map setting, compile option	Configure File setting
6	C31hex.cmd	Hex code generation option	

3.3.1 C31.C

```
// Circular Variable Init
void circular_init()


// Circular Data --> Liner Data convertion
int Data_Arrange( volatile int *FFTBufferIn )
int Data_Arrange_Stationary( volatile int *RAWBufferIn, volatile int *FFTBufferIn , int raw_ptr)


// Raw To FFT Converter
//            1. integer --> float
//            2. Window Data Converter
//              Window Flag : 1=window reference(trigger channel), 0=window response
//              Window : 0=time,1=time&window,2=spectra,3=time&spectra,4=avg spectra
//            3. FFT
//            4. float --> integer
//            5. FFTBufferIn : Raw Data , FFTToIEEE   : Process Data( FFT )
int RawToFFT( int block_start, volatile int *FFTBufferIn , volatile int *FFTToIEEE, int Channel )


// The Hanning command defines Hanning window for the filter.
// Filter : Low Pass Filter
// Equation :
//            w(n) = Window value at point n
//            N    = Filter Length
//            PI   = 3.14159....
//            w(k) = 0.54 - 0.46 COS( 2*PI * ( k/(N-1)) ) ,  k= 0 ~ N-1
int Hamming_Window()


// The Hanning command defines Hanning window for the filter.
// Filter : Low Pass Filter
// Equation
//            w(n) = Window value at point n
//            N    = Filter Length
//            PI   = 3.14159....
//            w(k) = 0.5 ( 1 - COS( 2*PI * ( k/(N-1)) ) ,  k= 0 ~ N-1
int Hanning_Window( int Filter_Len)
```

```
////////////////////////////////////////////////////////////////////////////////
// You Must enter a percentage in th Decay Rate % field. The number used is a percent of
// the ordinate range and dicatates the percent of signal dropoff at the frame midpoint.
// Equation :          x(n) * exp(-#*n)                    ; #- define unlimit
//                     n    = Exponential Decay
//                     #    = -(2/N) * log( n / 100 )
//                     N    = Filter Length
int Exponential_Window( int Filter_Len, double exp_variable )


// Trigger Channel Value -> except Trigger Point value, set 0 ( Clear )
int Rectangular_Window( volatile int *FFTBufferIn)


// Host --> Target Command Analysis & Processing
int Command_Analysis()


// Circular & Mode Variable Init.( Flag )
void ptr_init()


// Sin Table
int sin_table()


// Main Program
main()
```

3.3.2 M31BOOT.ASM

```
*******************************************************************************
* Interrupt sevice routinue( INT 0 )
* Host --> Target command receive interrupt
* Argment : None
* Return  : Command( ret_dat, interrupt flag )
*******************************************************************************
_mail_isr:

*******************************************************************************
* Interrupt sevice routinue( INT 2 )
* A/D Chip half interrupt
* Argment : None
* Return  : Command( interrupt flag )
*******************************************************************************
_receive_isr:

*******************************************************************************
* Main program
*******************************************************************************
start:

*******************************************************************************
* Interrupt occurs check ( INT 2 )
* Argment : None
* Return  : Command Data
*******************************************************************************
get4_mail:

*******************************************************************************
* Interrupt occurs check ( INT 2 )
* Argment : Command Data
* Return  : None
*******************************************************************************
put_mail:

*******************************************************************************
* Interrupt occurs check ( INT 2 )
* Argment : None
* Return  : Interrupt Flag
*******************************************************************************
ad_mem_move:
```

3.3.3 SUB.ASM

```
*************************************************************************
* Host --> Target Command Receive( INT0 assign )
* Argment : None
* Return  : Command( ret_dat )
* notice    : wait on until interrupt happens
*************************************************************************
_cget4_mail:

*************************************************************************
* Host --> Target Interrupt Flag Check
* Argment : None
* Return  : Interrupt Flag
*                          0 : no interrupt
*                          1 : interrupt occurs
*************************************************************************
_cget4_mail_flag:

*************************************************************************
* Timer 1 Install
* Argment : None
* Return  : None
*************************************************************************
_c3xtimer:

*************************************************************************
* Timer 1 Stop
* Argment : None
* Return  : None
*************************************************************************
_c3xtimer_reset:

*************************************************************************
* Timer 1 Setup
* Argment : Period Count
* Return  : None
*************************************************************************
_c3xtimer_init:

*************************************************************************
* FiFO 1K(512 * 2)  ping/pong
* FIFO --> Circular Memory
* Argment : None
* Return  : None
*************************************************************************
_cget_ad:
```

```
*******************************************************************************
* Trigger Level Check
* Argment : None
* Return  : Trigger Channel Data
*******************************************************************************
_cget_ad_ch0:

*******************************************************************************
* Memory Point Reset
* Argment : None
* Return  : None
*******************************************************************************
_cad_memory_reset:

*******************************************************************************
* A/D half Interrupt occurs check( INT2 assign)
* Argment : None
* Return  : Interrupt flag
*******************************************************************************
_cget_ad_flag:

*******************************************************************************
* FiFO Memory Reset
* Argment : None
* Return  : None
*******************************************************************************
_cad_init:

*******************************************************************************
* A/D Chip start( Data Acquisitio begin )
* Argment : None
* Return  : None
*******************************************************************************
_cad_start:

*******************************************************************************
* A/D Chip stop( Data Acquisitio stop )
* Argment : None
* Return  : None
*******************************************************************************
_cad_end:

*******************************************************************************
* Target --> Host Command trans
* Argment : Command
* Return  : None
*******************************************************************************
_cput_mail:
```

```
*******************************************************************************
* Channel 0 gain control
* Argment : gain data
* Return  : None
*******************************************************************************
_gain_ctrl_ch0:

*******************************************************************************
* Channel 1 gain control
* Argment : gain data
* Return  : None
*******************************************************************************
_gain_ctrl_ch1:

*******************************************************************************
* Channel 2 gain control
* Argment : gain data
* Return  : None
*******************************************************************************
_gain_ctrl_ch2:

*******************************************************************************
* Channel 3 gain control
* Argment : gain data
* Return  : None
*******************************************************************************
_gain_ctrl_ch3:
```

3.3.4 FFT.ASM

```
*****************************************************************************
* FILENAME          : ffft_rl.asm
*
*
* SYNOPSIS:         int             ffft_rl( FFT_SIZE, LOG_SIZE, SOURCE_ADDR, DEST_ADDR,
*                                            SINE_TABLE, BIT_REVERSE );
*
*                                   int             FFT_SIZE        ; 64, 128, 256, 512, 1024, ...
*                                   int             LOG_SIZE        ; 6,  7,  8,  9,  10, ...
*                                   float           *SOURCE_ADDR    ; Points to location of source data.
*                                   float           *DEST_ADDR      ; Points to where data will be
*                                                                   ; operated on and stored.
*                                   float           *SINE_TABLE     ; Points to the SIN/COS table.
*                                   int             BIT_REVERSE     ; =  0, Bit Reversing is disabled.
*                                                                   ; <> 0, Bit Reversing is enabled.
*
*                                   NOTE:           1) If SOURCE_ADDR = DEST_ADDR, then in place bit
*                                                     reversing is performed, if enabled (more
*                                                     processor intensive).
*                                                   2) FFT_SIZE must be >= 64 (this is not checked).
*
* DESCRIPTION:      Generic function to do a radix-2 FFT computation on the C30.
*                                   The data array is FFT_SIZE-long with only real data. The
*                                   output is stored in the same locations with real and imaginary
*                                   points R and I as follows:
*
*                                   DEST_ADDR[0]                    -> R(0)
*                                                                      R(1)
*                                                                      R(2)
*                                                                      R(3)
*                                                                      .
*                                                                      .
*                                                                      R(FFT_SIZE/2)
*                                                                      I(FFT_SIZE/2 - 1)
*                                                                      .
*                                                                      .
*                                                                      I(2)
*                                   DEST_ADDR[FFT_SIZE - 1]         -> I(1)
*
*                                   The program is based on the FORTRAN program in the paper
*                                   by Sorensen et al., June 1987 issue of Trans. on ASSP.
*
*                                   Bit reversal is optionally implemented at the beginning
*                                   of the function.
*
```

```
*                                       The sine/cosine table for the twiddle factors is expected
*                                       to be supplied in the following format:
*
*                                       SINE_TABLE[0]                   -> sin(0*2*pi/FFT_SIZE)
*                                                                          sin(1*2*pi/FFT_SIZE)
*                                                                   .
*                                                                      .
*                                                                          sin((FFT_SIZE/2 -2)*2*pi/FFT_SIZE)
*                                       SINE_TABLE[FFT_SIZE/2 - 1] -> sin((FFT_SIZE/2 -1)*2*pi/FFT_SIZE)
*
*                                       NOTE: The table is the first half period of a sine wave.
*
*                                       Stack structure upon call:
*
*                                       +------------------+
*                                       -FP(7)  | BIT_REVERSE    |
*                                       -FP(6)  | SINE_TABLE     |
*                                       -FP(5)  | DEST_ADDR      |
*                                       -FP(4)  | SOURCE_ADDR    |
*                                       -FP(3)  | LOG_SIZE       |
*                                       -FP(2)  | FFT_SIZE       |
*                                       -FP(1)  | return addr    |
*                                       -FP(0)  | old FP         |
*                                        +------------------+
*
*************************************************************************************
*
*                                       NOTE:           Calling C program can be compiled using either large
*                                                       or small model.
*
*                                       WARNING: DP initialised only once in the program. Be wary
*                                                       with interrupt service routines. Make sure interrupt
*                                                       service routines save the DP pointer.
*
*                                       WARNING: The DEST_ADDR must be aligned such that the first
*                                                       LOG_SIZE bits are zero (this is not checked by the
*                                                       program).
*
*************************************************************************************
*
* REGISTERS USED: R0, R1, R2, R3, R4, R5, R6, R7
*                                         AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7
*                                         IR0, IR1
*                                         RC, RS, RE
*                                         DP
*
* MEMORY REQUIREMENTS:                  Program = 405 Words (approximately)
*                                       Data    =   7 Words
*                                       Stack   =  12 Words
*
*************************************************************************************
*
* BENCHMARKS:   Assumptions             - Program in RAM0
*                                       - Reserved data in RAM0
*                                       - Stack on Primary/Expansion Bus RAM
*                                       - Sine/Cosine tables in RAM0
*                                       - Processing and data destination in RAM1.
*                                       - Primary/Expansion Bus RAM, 0 wait state.
*
*                       FFT Size        Bit Reversing       Data Source         Cycles(C30)
*                       --------        -------------       -----------         -----------
*                       1024            OFF                 RAM1                19816 approx.
*Note                   : This number does not include the C callable overheads.
*                         Add 57 cycles for these overheads.
*
```

3.4 Library Reference

• **PCI Function**

int cput_mail();
int cget4_mail();
int cget4_mail_flag();

• **A/D Function**

int cad_init();
int cad_start();
int cad_end();
int cget_ad();
int cget_ad_ch0();
int cget_ad_ch1();
int cget_ad_ch2();
int cget_ad_ch3();
int cget_ad_flag();
int cad_memory_reset();

• **Gain Control Function**

int gain_ctrl_ch0();
int gain_ctrl_ch1();
int gain_ctrl_ch2();
int gain_ctrl_ch3();

Timer Install & Sampling Time Install Function

int c3xtimer();
int c3xtimer_reset();
int c3xtimer_init();

• **FFT Function**

int ffft_rl();

• **Window Function**

int Hamming_Window( )
int Hanning_Window( )
int Exponential_Window(int Window_Flag)
int Rectangular_Window(int Window_Flag)

int cput_mail(int Command)

[Description] Host → Target Command transfer

[Argument] 1. PASS : Command pass
2. FALSE : Command fail

[Return Value] None

[Usage] 1. cput_mail(PASS); /* Command OK ! */
2. cput_mail(FALSE); /* Command NOK ! */

int cget4_mail()

[Description] If Host → Target Command is available.
If available, Command Variable store and set Flag.
< Note: this function wait on, until Interrupt occur. >

[Argument] None

[Return Value] Command Data

[Usage]

```
command = cget4_mail();          // command read
switch( command ) {
                case 0x10 :
                                 cad_start();
                                 break;
                default :
                                 break;
}
```

int cget4_mail_flag()

[Description] Host → Target tx mail flag Return

[Argument] None

[Return Value]
- 0 : No Interrupt occurrence
- 1 : Host → Target Communication Interrupt occurrence

[Usage]
```
mail_flag = cget4_mail_flag();        // mail flag check
if ( mail_flag != 0 ) {
                    Command_Analysis() ;
}
```

int cad_init()

[Description] A/D FIFO Memory Reset

[Argument] None

[Return Value]
- 0 : Memory Reset OK
- 1 : Memory Reset Error

[Usage]
```
if ( cad_init() == PASS ) {
                    cput_mail( PASS);   /*  Command  OK ! */
}
else {
                    cput_mail( FALSE);   /*  Command  NOK ! */
}
```

int cad_start()

[Description] A/D Chip Start Signal Output

[Argument] None

[Return Value] None

[Usage] cad_start();

int cad_end()

[Description] A/D Chip end Signal Output

[Argument] None

[Return Value] None

[Usage] cad_end();

int cget_ad()

[Description] Acquisition Data storage

[Argument] None

[Return Value] None

[Usage] cad_ad();

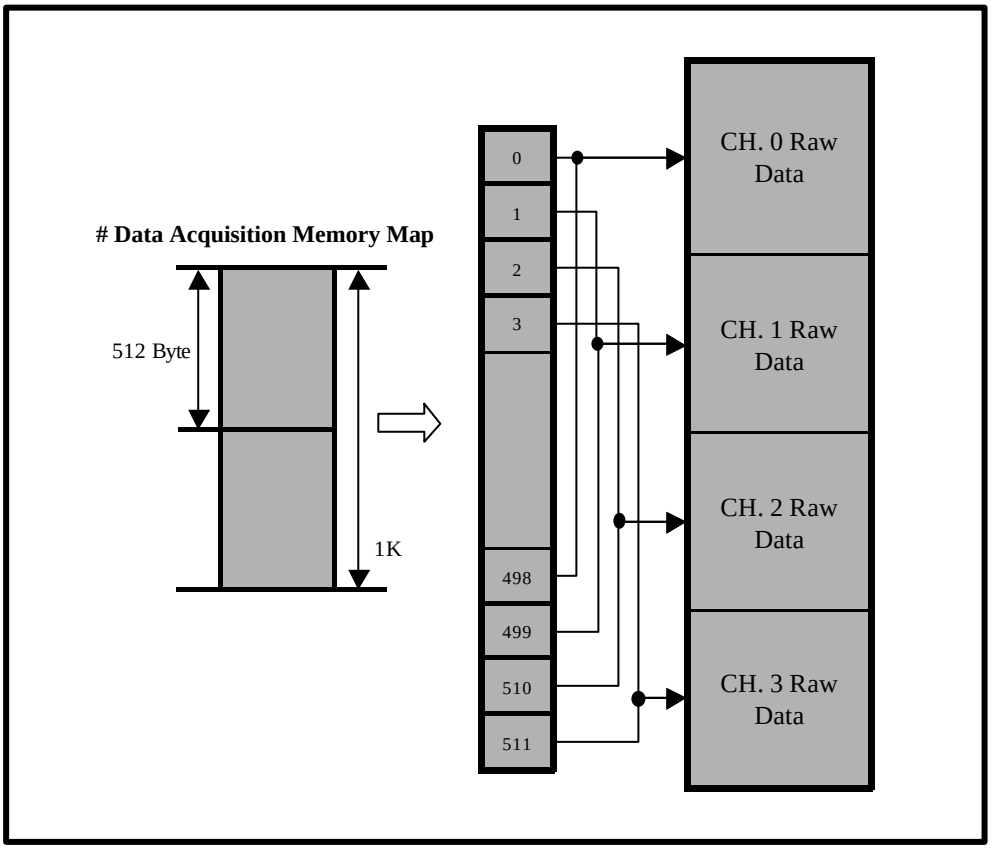


Figure 3-1 Acquisition Data storage

int cget_ad_ch0( )

[Description] Channel 0 Current Point Data Return

[Argument] None

[Return Value] Memory Data

[Usage] cget_ad_ch0( ) ;

int cget_ad_ch1( ), int cget_ad_ch2( ), int cget_ad_ch3( ) is equal.

int cget_ad_flag()

[Description] A/D Chip data quantity is Half or not half

[Argument] None

[Return Value]
- 0 : No Interrupt
- 1 : FIFO Half Interrupt Occur

[Usage]

```
ad_flag = cget_ad_flag();
if ( ad_flag != 0 ) {
                cget_ad();
            .
}
```

int cad_memory_reset()

[Description] Channel 0 ~ 3 Data Point Reset

[Argument] None

[Return Value] None

[Usage] cad_memory_reset();

int gain_ctrl_ch0(Int Command)

[Description] Channel gain control

[Argument] None

[Return Value] None

[Usage]
```
if  (gain_ctrl;_ch0( command ) == PASS )  {
                    cput_mail( PASS);    /*   Command   OK ! */
}
else {
                    cput_mail( FALSE);    /*   Command   NOK ! */
}
```

Table 3-3 Command Define

	a high position Bit(Channel)				a subordinate position Bit(magnification)			
Bit	7	6	5	4	3	2	1	0
Define	CH. 3	CH. 2	CH. 1	CH. 0	10	5	2	0

• Example)

Command : 0x12

gain_ctri_ch0(Command)

. Description : 0x 12

Binary : 0001 0010

Channel : 0001 = 0 Channel

magnification : 0010 = 2 magnification

int c3xtimer()

[Description] Timer Start

[Argument] None

[Return Value] None

[Usage]
```
if  (c3xtimer() == PASS )  {
                    cput_mail( PASS);    /*   Command   OK ! */
}
else {
                    cput_mail( FALSE);    /*   Command   NOK ! */
}
```

int c3xtimer_reset()

[Description] Timer Stop

[Argument] None

[Return Value] None

[Usage]
```
if  (c3xtimer_reset() == PASS )  {
                    cput_mail( PASS);    /*   Command   OK ! */
}
else {
                    cput_mail( FALSE);    /*   Command   NOK ! */
}
```

int c3xtimer_init()

[Description] Timer Setup

[Argument] None

[Return Value] None

[Usage]
```
if  (c3xtimer_init() == PASS )  {
                    cput_mail( PASS);    /*   Command   OK ! */
}
else {
                    cput_mail( FALSE);    /*   Command   NOK ! */
}
```

int Hamming_Window()

[Description] Raw → Window Data

[Argument] None

[Return Value] None

[Usage] Hamming_Window()

int Hanning_Window()

[Description] Raw → Window Data

[Argument] None

[Return Value] None

[Usage] Hanning_Window()

int Exponential_Window(int Window_Flag)

[Description] Raw → Window Data

[Argument] None

[Return Value] None

[Usage] Exponential_Window(int Window_Flag)
# Window_Flag : 0 = Response Channel
1 = Reference Channel

int Rectangular_Window(int Window_Flag)

[Description] Raw → Window Data

[Argument] None

[Return Value] None

[Usage] Rectangular_Window(int Window_Flag)

Window_Flag : 0 = Response Channel

1 = Reference Channel

Chapter 4
Software Description

This chapter describes software of the DMP1162B.

Chapter contains

4.1 Description

Software consists of windows driver(VxD), dynamic linking library (DLL) and application software. Application software is called as "DmeXpress" in this manual. Application software shall be developed with the environment of the Visual C++ 6.0, Microsoft. The source codes of the application programs are provided in the CD.

The development environment for windows driver(VxD) is referred to chapter 2.1.2, Development Environment. The simple example of the windows driver(VxD) is described. The development of windows driver(VxD) shall be referred to DDK of Microsoft, PCI SDK of PLX Technology.

The simple example for Dynamic Linking Library (DLL) is described.

4.1.1 Overview

This manual describes how to setup the DmeXpress system to accept the incoming data signal(s), to specify the signal processing to be performed and the required display format and how to save, recall, delete, and download data. The manual also explains basic operations related to the features and functions available within the application software. Figure 4-1 shows the process of acquiring data through the DmeXpress system.

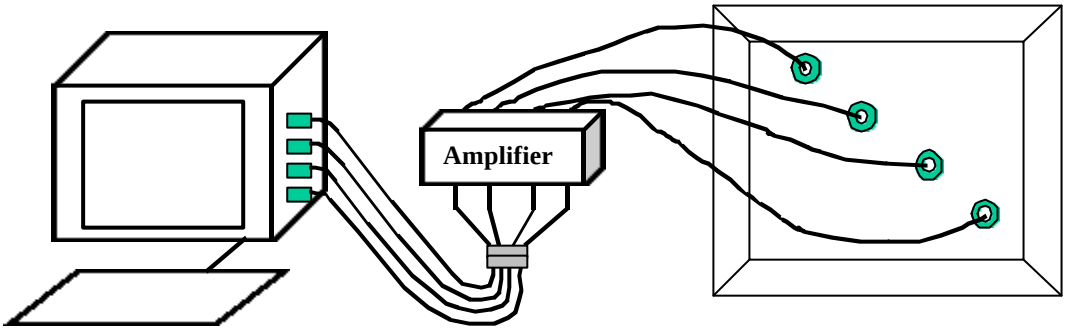


Figure 4-1 Acquiring Data through DmeXpress

This application software is used to:

- set up the sampling condition options
- define the input channels and the related options
- preview incoming data
- acquire signal data
- perform data analyses
- etc

4.1.2 Concepts of the Data Acquisition With DmeXpress Software

There are some concepts that are necessary for being defined. This chapter discusses:

- input channels
- real time displays
- Engineering units
- Menu forms
- file management
- display management

Input Channels

The total number of DmeXpress input channels is 4. Each channel is defined in the channel setup menu to be connected with the corresponding physical channel. A channel refers to the setup information associated with a particular transducer. Part of the channels could optionally be selected to be active or inactive. Active channels are the channels you select through the channel setup menu, and inactive channels are those you have not chosen before measurements.

Real Time Displays

Real time displays are used to quickly scan active input channels in various formats.

There are two kinds of formats in the window displays. One is single column layout, and the other is double column layout. The single column layout displays the incoming time signal data corresponding to the active channels in a single column. The double column layout is used for displaying incoming time data and other data in different format such as windowed time signal, spectra, etc.

Engineering units(EU)

This is used to control data scaling and ordinate labelling operations when required. This unit is utilized in conjunction with SENSITIVITY in the ‘ Input Channel’ and UNITS in the ‘ Display Option’ menu forms. The ‘ Sensitivity’ parameter let you convert the input voltage into your own EU so that it may become any kind of physical unit used in Metric or English.

Menu Forms

Every menu that needs adjusting more parameters or subsidiary options has its own menu form. The form is a special window having several entries to control setting parameters. The ellipses(..) mark after a command menu means it has a form. The control parameters in the form lets you find default or current settings and lets you also change them. The selectable parameters contained in the form has various kinds of types such as control buttons, predefined settings, status information, hard ware dependent values, etc.

File Management

It is necessary to manage the test data acquired or processed to be used efficiently when needed. To achieve this, the File menu is prepared to save the data in a formatted form and reproduce it for a later use. This menu has a variety of file handling commands such as Save, Save as, Export, Import. Also the file saving in another way is possible by using the toggle switch box in the Measurements menu form. The display format of the recalled data from the saved file or the imported data from other external software is specified in the Display Option menu.

Display Management

The display methods are necessary during data acquisition process and when data reproducing for the data retrieving and off-line data processing. The display parameter settings related to the data acquisition process are defined in the Real Time Display form in the Overall Setup menu, and the display options for the off-line processing are specified in the Display Option form of the Display Setup menu. Once the display options are specified, they works anywhere in the software.

4.2 Set Up

4.2.1 Introduction to Setup

There are three different Setup sub-menus in the main menu. Figure 4-2 shows the display screen of the Setup menu. This menu includes:

- Overall Setup
- Channel Setup
- Display Setup

The subsequent chapters of this manual describes the various settings required to acquire, process and display data using the application software.

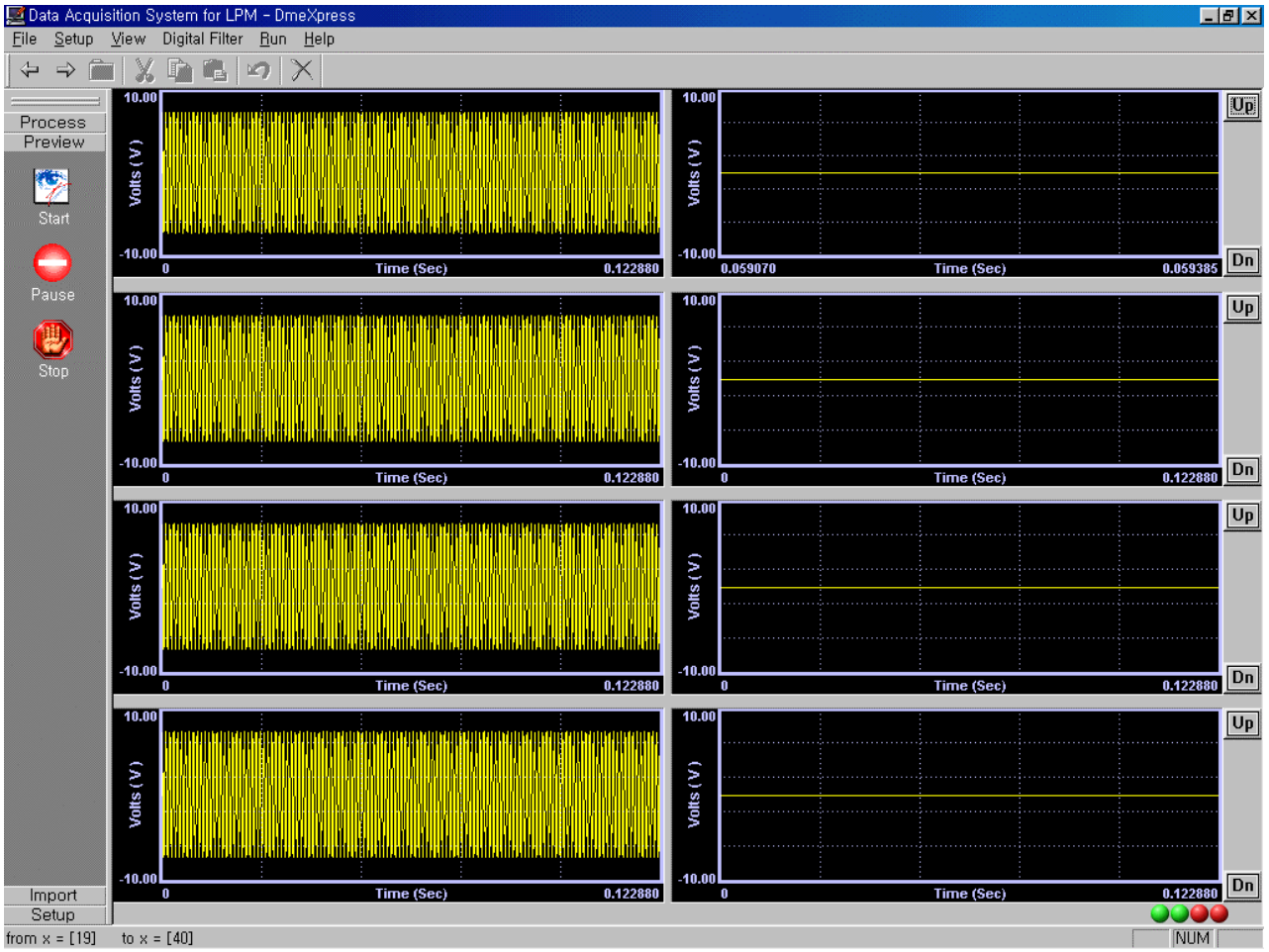


Figure 4-2. Setup menu

4.2.2 Overall Setup

The Overall Setup menu defines general measurement conditions to characterize sampling and trigger conditions, real time display options, windowing methods, averaging and overlapping parameters, and the results type.

This chapter describes how to set:

- real time display options
- sampling conditions
- trigger events
- windowing and averaging options
- measurements options

General

General form provides for applying window and averaging to your acquiring data during data acquisition. Figure 4-3 shows the general form of the Overall Setup menu.

The window types available to this version is as follows:

- Rectangular
- Hanning
- Hamming
- Exponential

The windows are applied to both reference and response channels separately.

If you have an impact signal as an input and thus select a rectangular window, then you must define the width of the window as a percentage of the frame length in the Width % field.

The exponential window also needs a Decay Rate % field. The number used in the field indicates a percent of the amplitude at the frame midpoint to the amplitude at the beginning of the frame.

When you perform the frequency domain data analysis with a stationary signal, more than one frame of data at a time will often be needed to reduce the noise effect on the spectra. The Frames per Avg field specifies the number of frames used for averaging when you acquire data. Overlapping is a process by which a current frame data includes a percent of previously processed data. The Overlap Percent % field gives 4 kinds of overlap percentage value such as 0%, 25%, 50%, and 75%.

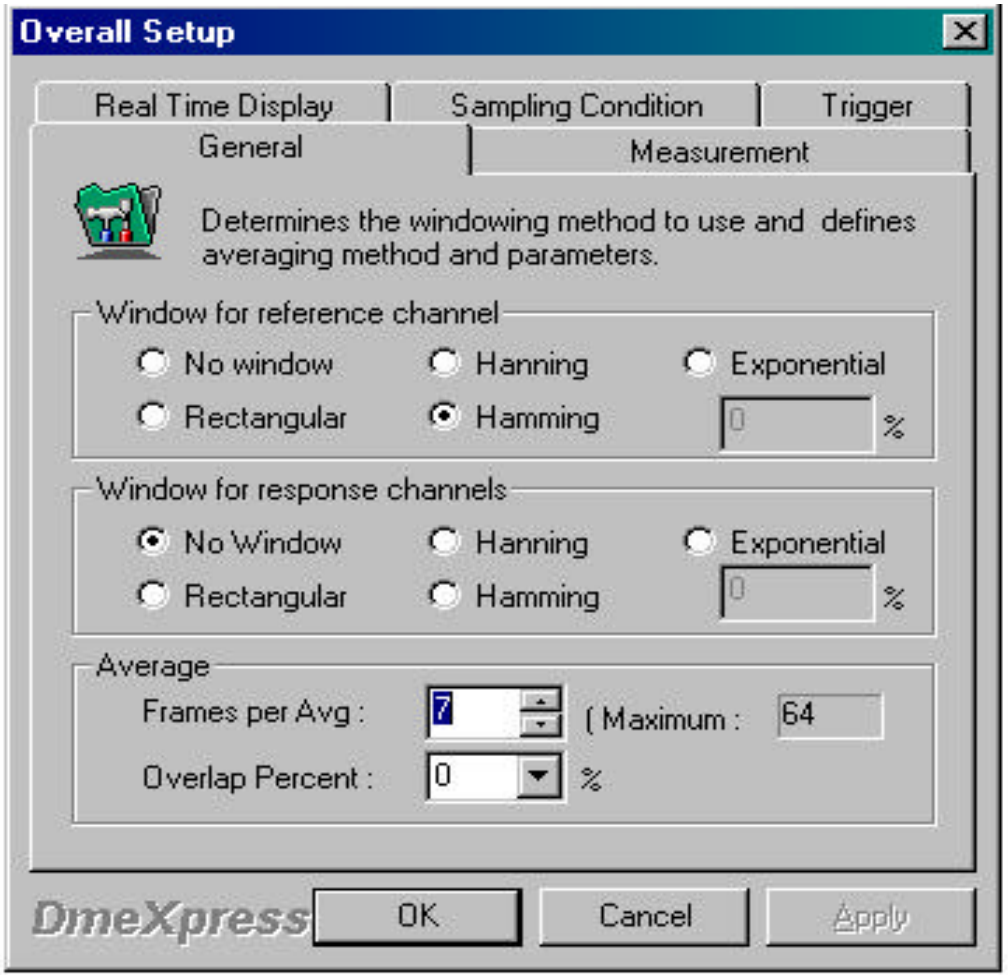


Figure 4-3. Overall setup

Sampling Condition

The Sampling Condition form makes you define the sampling characteristics for both time and frequency domain data analyses. There are two fields on the left side of the form that need user input based on Nyquist sampling theorem. The values of the right side fields are status information automatically calculated. Thus, if you change a value in one field in the form, other field values are changed automatically. Figure 4-4 illustrates the Sampling Condition form. Every field is maintained up to date with each value you modify on the page. For example, if you enter maximum frequency and frame size in the form to 200 Hz and 2048, respectively, the values of other parameters including sampling frequency, frequency resolution and frame length on the right side automatically changes to 512 Hz, 0.25 Hz, and 4.0 sec. The actual data sampling rate for each channel has already been adjusted to be 2.56 times the maximum frequency bandwidth.

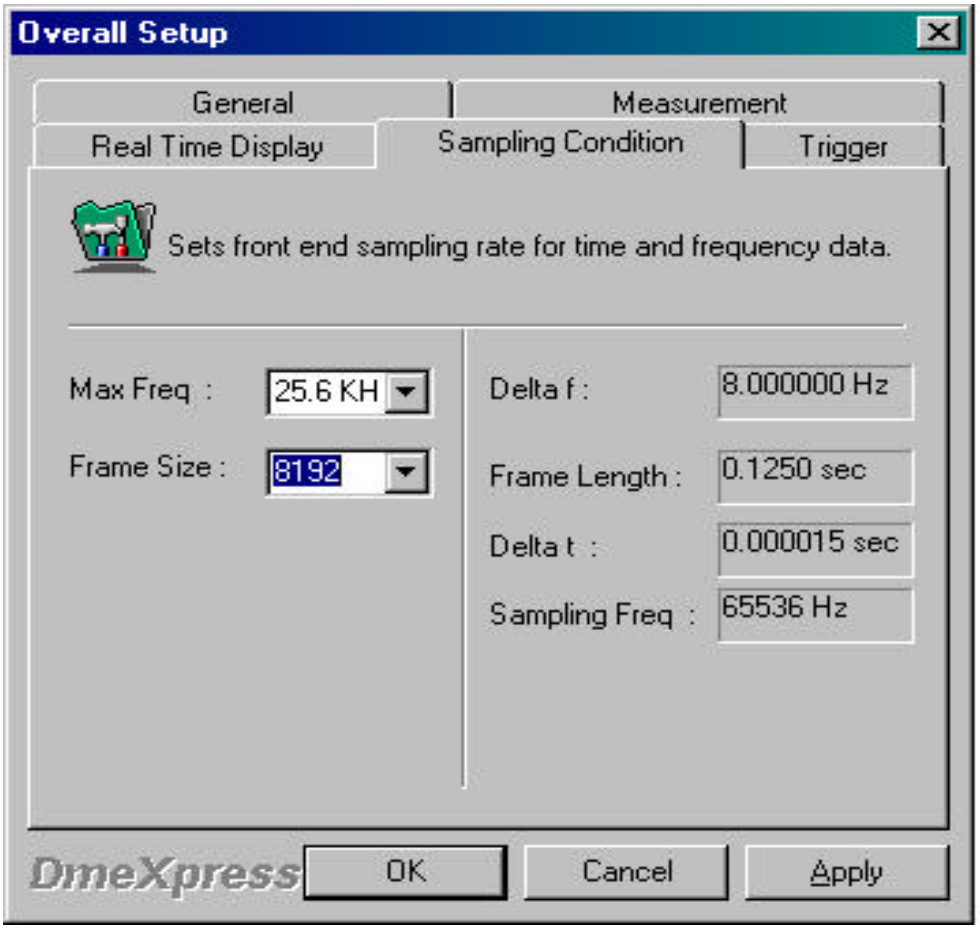


Figure 4-4. Sampling Condition

Trigger Events

There are two methods available for triggering which defines when to begin acquiring sampling data(see Figure 4-5):

- free run
- frame by frame

In case of free run, data collection starts when you select the Run command from the main menu. There is no need to define any options. If you select Frame by Frame, a trigger condition will be checked as every data frame is acquired.

When the trigger method is determined as Frame by Frame, you must specify the Trigger Channel, Level %, Slope, and Pretrigger values. The frame by frame trigger method implies the discontinuous sampling which is generally applied to a transient signal processing.

Trigger Channel means the input channel for the trigger source. Level % indicates the level above the point where the software should begin acquiring data. The Slope defines the slope of the signal used to

initiate triggering. The value may be either positive or negative. The Pretrigger field enables negative delay between the triggering event and the beginning of frame data. This effect works on the whole channels simultaneously. For example, by entering 5 % in the Pretrigger field under 1024 frame size, you can find that there esist 51 sample points prior to the trigger point. The Trigger Bell switch makes you listen to beep sound when the trigger event becomes ready to acquire data on a frame by frame basis.

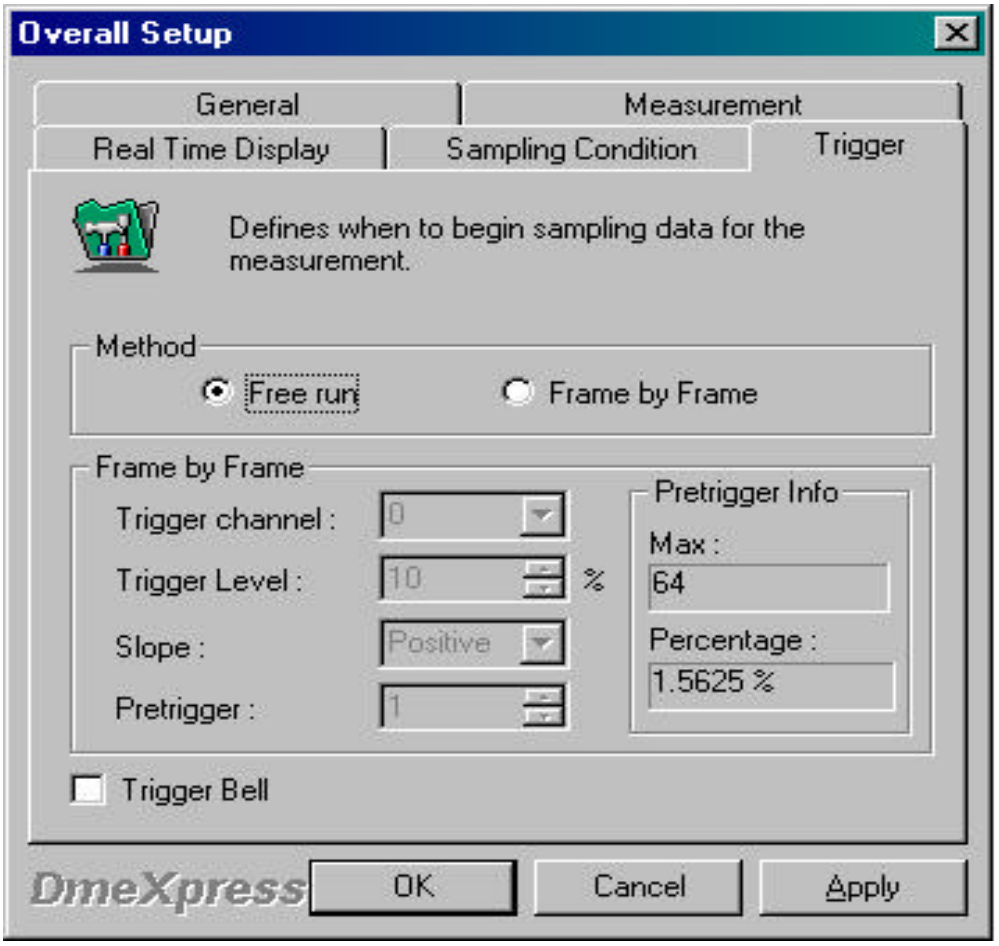


Figure 4-5. Trigger

Real Time Display

The Real Time Display form(Figure 4-6) provides the control information of the types and the number of acquisition displays during real time signal processing. Acquisition Monitor switch gives the format you want to use to display data being acquired. There are six kinds of types to display acquiring data such as Time, Time & Windowed, Spectra, Time & Spectra, Time & Avg Spectra. If you select Time, time history data of input channels are displayed in single column format. If you select Time & Windowed, double columns of data per channel are displayed. The time history data of the channel are in column one and the

windowed time history data are in column two. Similarly if you select Spectra or Time & Spectra or Time & Avg Spectra, the corresponding data are displayed accordingly in column one or column two formats. The Y Axis Scale switch determines the scaling of the ordinate of the real time display. The scaling has three different options, namely Linear, Log, and dB. The dB scale is defined as 10 times of logarithmic value of the data with base 10. The real time data may be displayed directly in Volts or in EU(Engineering Units) by using the Display Unit field option. The EU can become any physical units through the sensitivity defined in the Channel Setup menu.

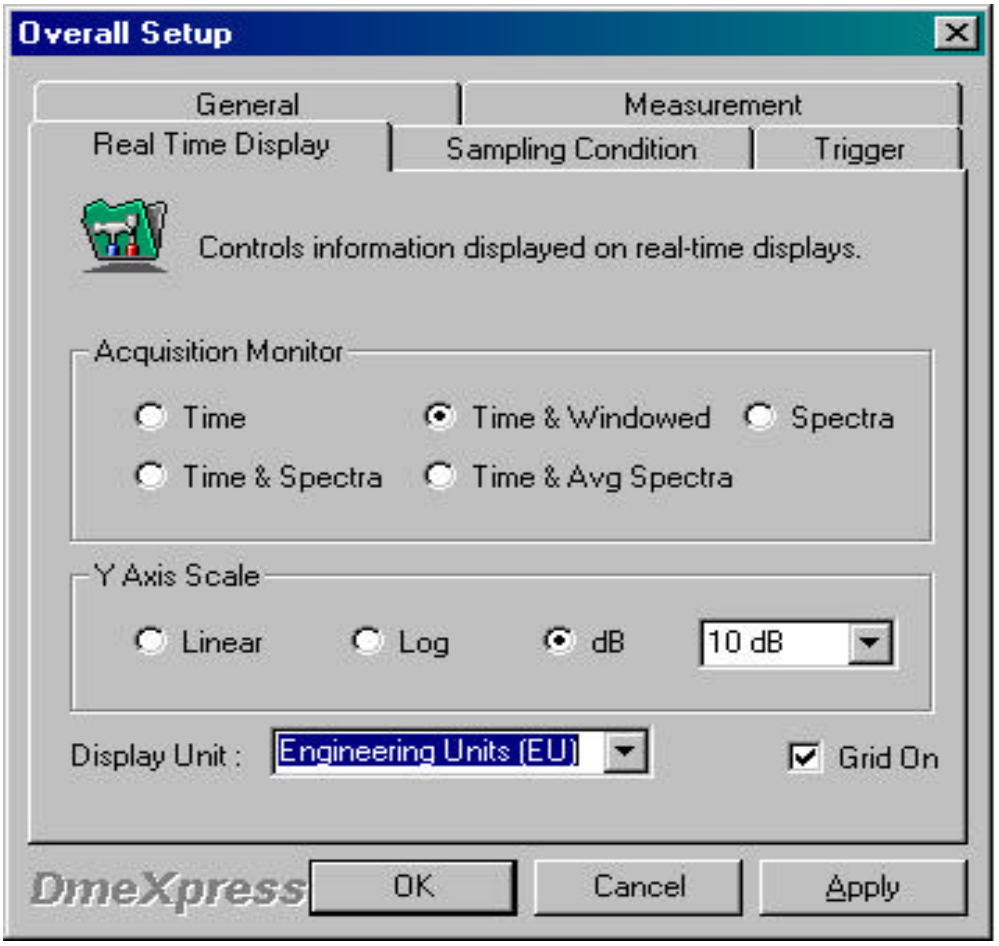


Figure 4-6. Real Time Display

Measurements

The Measurement form(see Figure 4-7) defines the type of data to be collected. If the previous options related to Overall Setup are completely set, it is ready to preview and acquire the incoming data. When the data is previewed, it is not necessary to make any settings on the gathering data. However, in order to store the processed result, you must request the type of the results to be saved using the Measurements

form. This form has five different Write Switches. Four of them are to save frequency domain results, and the other to collect time domain results. Every data file to be saved has its own file extension according to a data type. Among the frequency domain results, the Auto Spectra and the magnitude of the Cross Spectra & FRF may have three kinds of units such as PS(Power Spectra), PSD(Power Spectral Density), and ESD(Energy Spectral Density) which are specified in the Spectrum Unit field. The FRF Method field enables you to choose different type of frequency response function between H1 and H2 whose definitions are referred to the ' Random Data – Analysis and Measurement Procedures written by Bendat & Piersol, 1991.

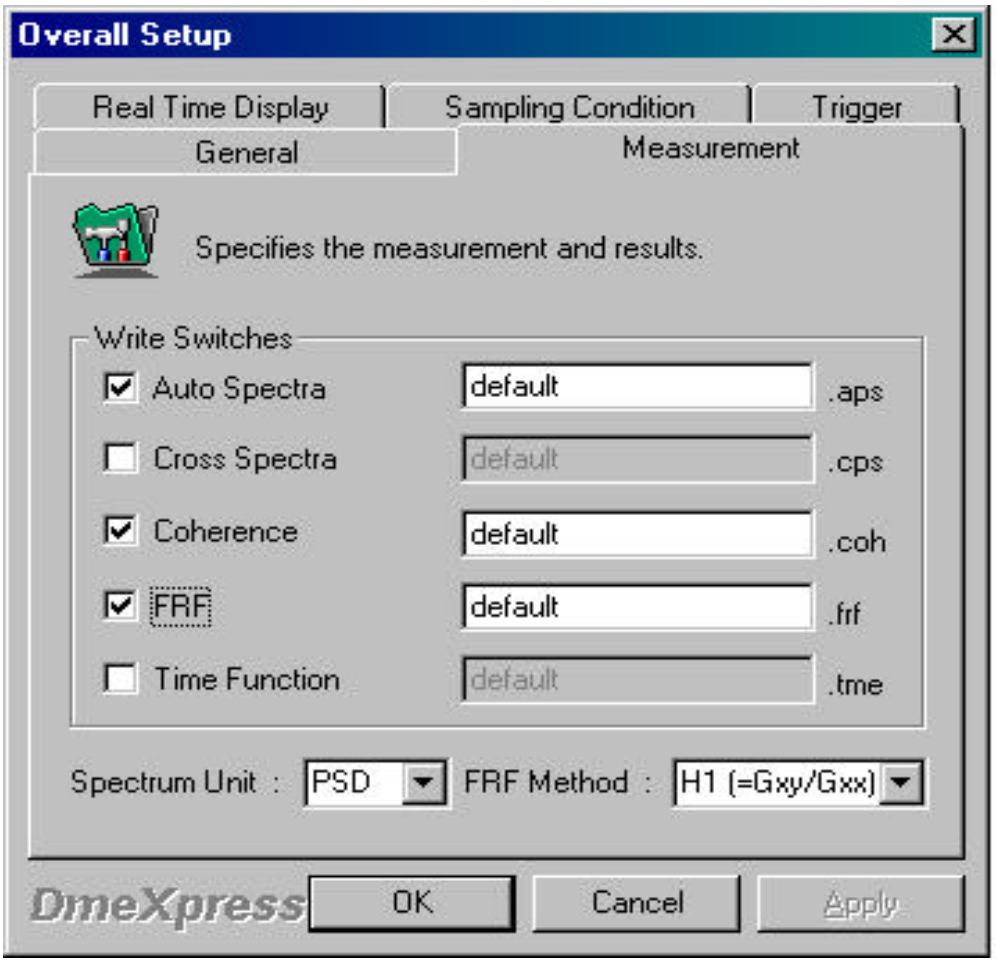


Figure 4-7. Measurements

4.2.3 Channel Setup

The primary function of the Channel Setup menu is to identify the reference channel to which the cross spectra and frequency response function are referred. The General field does the above task(Figure 4-8).

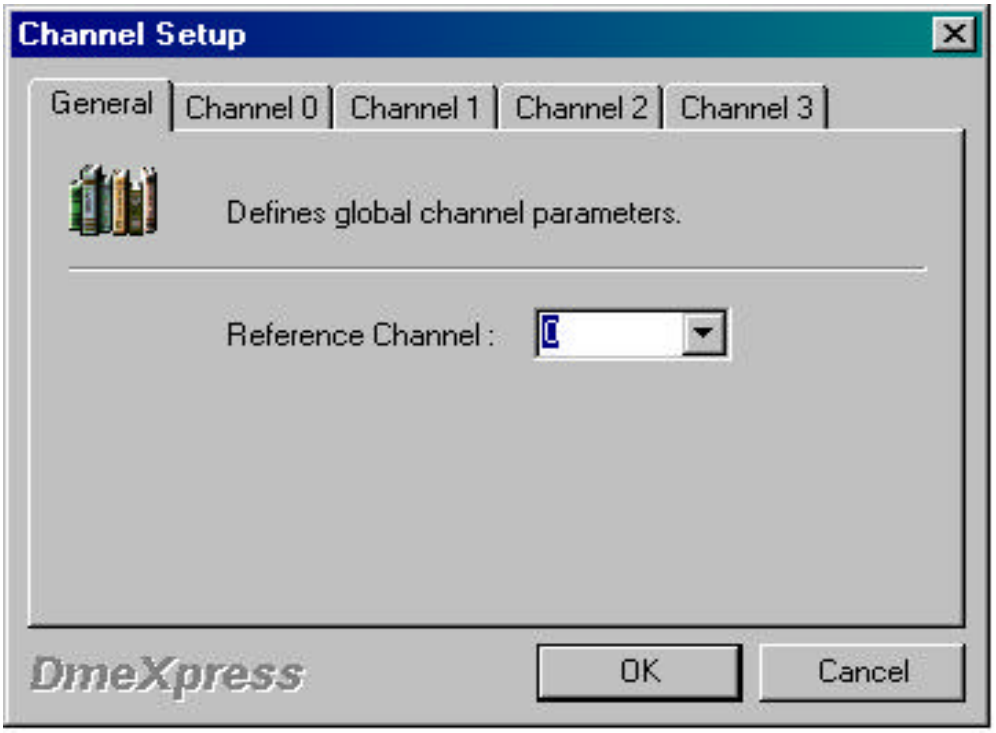


Figure 4-8. General

The four different channel fields are prepared to connect physical channels, to set input ranges, and to associate the transducers. As shown in Figure 4-9, the DSP Gain field is prepared to match the gain setting on the amplifier or signal conditioner. The Sensitivity is similar to the scale factor. It converts the voltage data into the physical data expressed by engineering units(EU). For example, if you have 1 Volt input voltage and the Sensitivity is 100 mV/g (g is a gravity unit in 9.81 m/s^2), then the physical input scale becomes 10 g(1000 mV divided by 100 mV/g). The Range field is given to manually set the maximum display range of the channel. For example, if you set 10 Volt in the field, the range of amplitude display for the channel is adjusted to ±10 Volt. The Coupling field is also provided to choose AC or DC coupling. The AC coupling option utilizes an AC filter which cuts off the signal data at somewhere low frequency. The DC coupling means no use of the AC filter. Thus a mean value of the data can be displayed only in the DC coupling option.

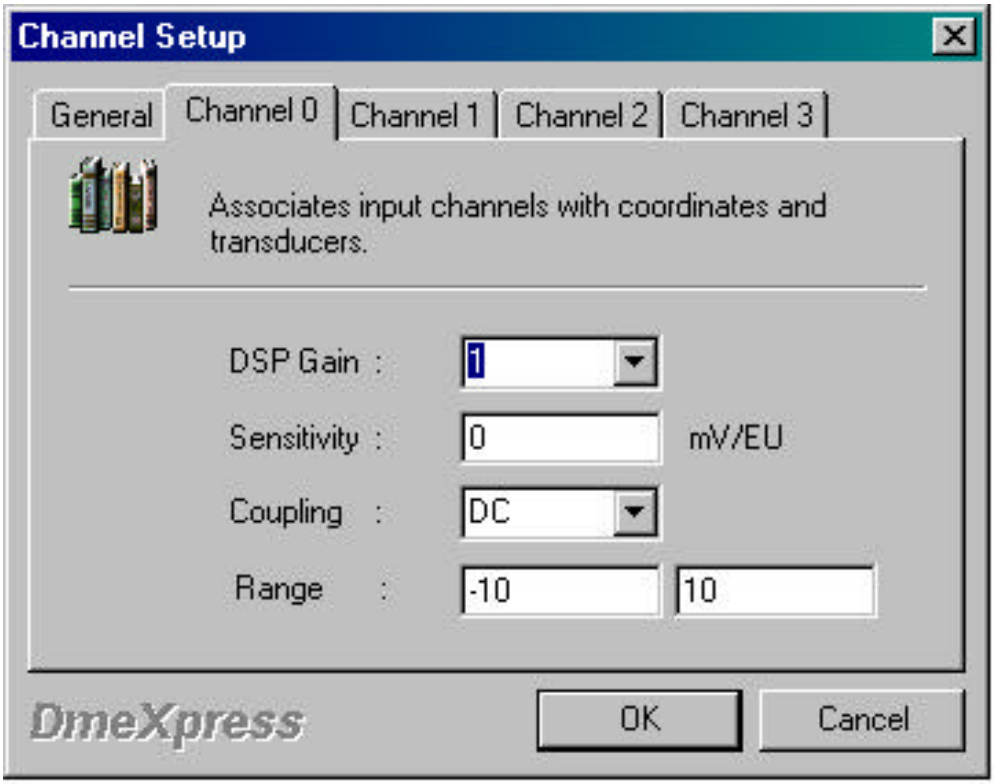


Figure 4-9. Channel Setup

4.2.4 Display Setup

The Setup controls the off-line display format for the data being imported. The Display Options form is shown in Figure 4-10. The Y Scale field allows you to choose one of the ordinate scaling types such as Linear, Log, dB 10, or dB 20. If the Auto Scale switch is checked, the input range is automatically set based on the level of the incoming signal for optimal viewing. Also the Grid On option generates the grid lines on the display screen.

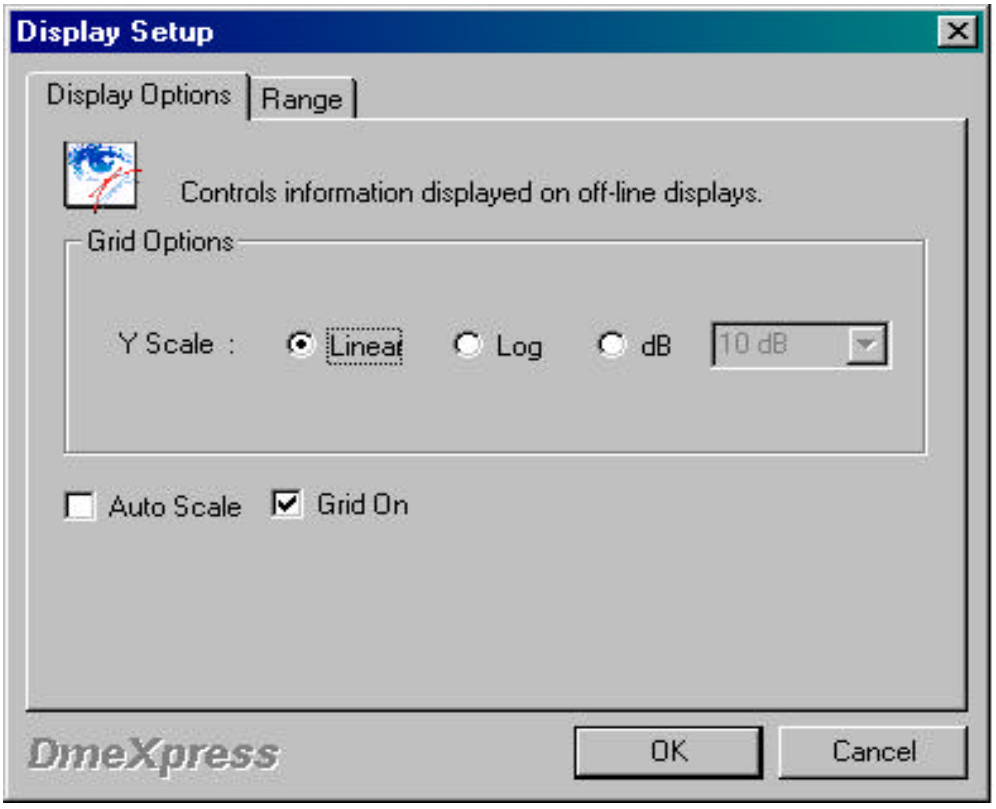


Figure 4-10. Display Options

The Range form(Figure 4-11) specifies the input range per channel manually. It is given two types, that is for time data and for spectral data. Also each range field may have both upper and lower limits separately.

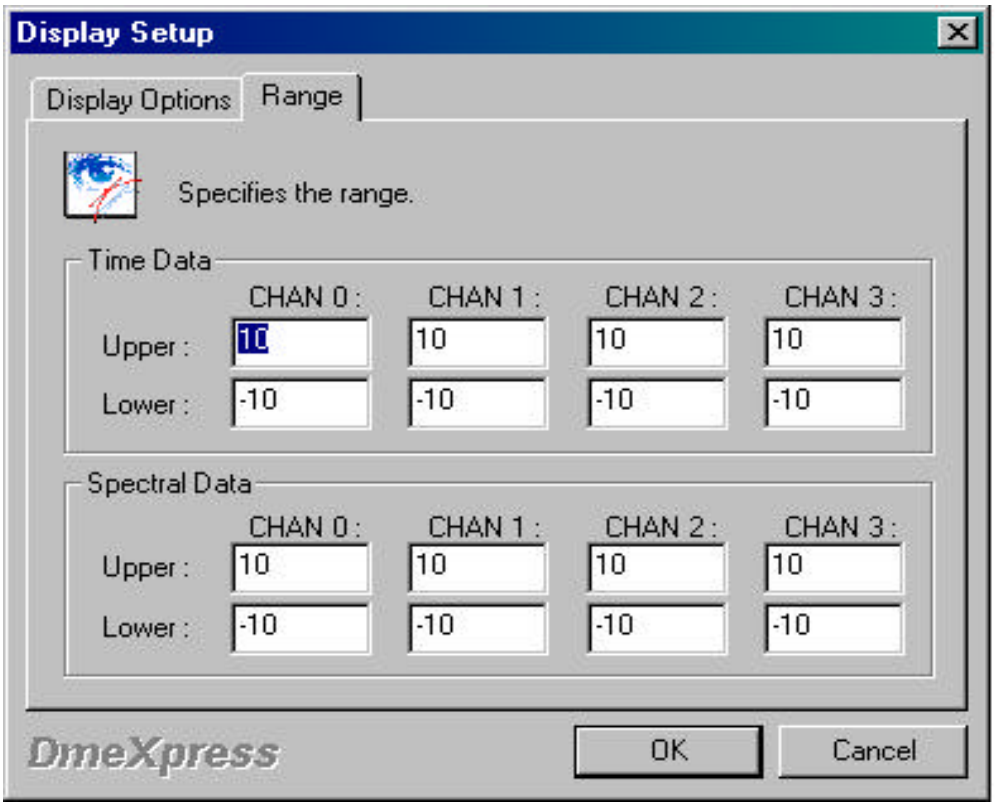


Figure 4- 11 Range

4.3 . Measurements

4.3.1 Introduction to Measurements

After the input source and the related setup options for the data acquisition have been established, you are ready to begin acquiring signal data. There are three main menus related to the data measurements:

- View
- Digital Filter
- Run

The View menu provides a previewing capability that allows you preview the input data and modify the setup options optimally before running the data measurement task. The Digital Filtering menu has capability of performing FIR(Finite Impulse Response) & IIR(Infinite Impulse Filter) filter works. The Run menu performs the actual measurement process using the information that you have defined in the Overall Setup and Channel Setup options. The subsequent chapters discusses how to preview and run the signals.

4.3.2 Preview

The trees of the View menu is illustrated in Figure 4-12. The menu allows you to scan the incoming data based on the selection of the following submenus:

- Time
- Time & Windowed
- Spectra
- Time & Spectra

Based on the above windows, you can adjust the input channel range optimally before running the data measurements. If you select Time, time history of incoming data per channel is displayed in single column format. If you select Time & Windows, both time history and its windowed time history data are simultaneously displayed in double columns format. If you select Spectra, the instantaneous spectrum of the time history data per channel is displayed in single column format. If you select Time & Spectra, both time history and corresponding spectrum data are simultaneously displayed in double columns format.

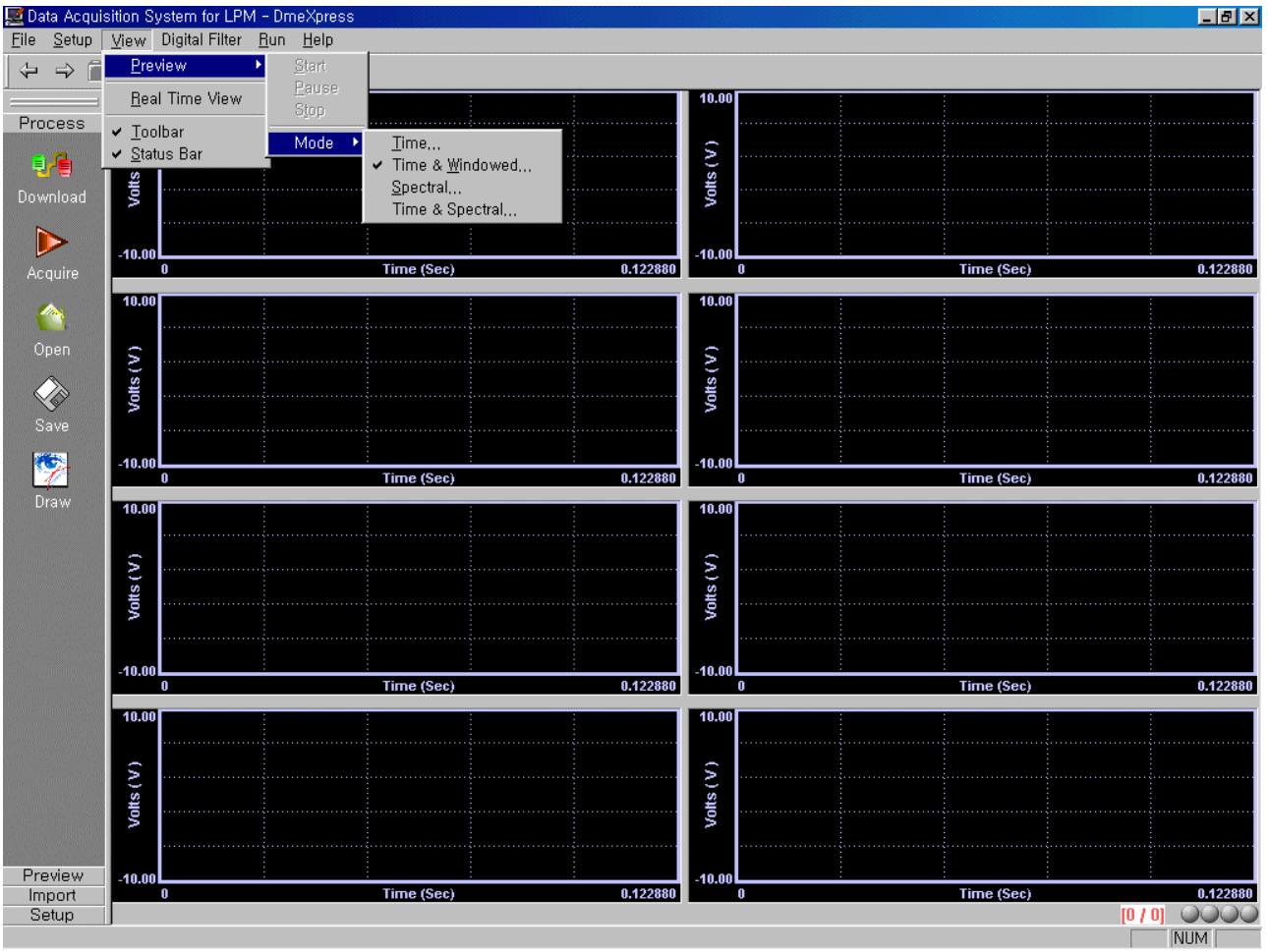


Figure 4-12. View

4.3.3 Digital Filter

The tree of the Digital Filter menu is illustrated in Figure 4-13. The menu allows you to perform digital filtering in the time domain. There are two kinds of popular digital filters such as FIR and IIR. The theory to digital filter technique is described in various kinds of signal processing text books. This menu consists of the following submenus:

- Apply Filter
- FIR Setup
- IIR Setup
- Open a file to be filtered …

Each filter has 4 different kinds such as High pass, Low pass, Band Pass, and Band Stop filters whose input parameters are defined in each Setup form(FIR Setup & IIR Setup). Figure 4-14 and Figure 4-15 shows the setting parameters of FIR and IIR filters.

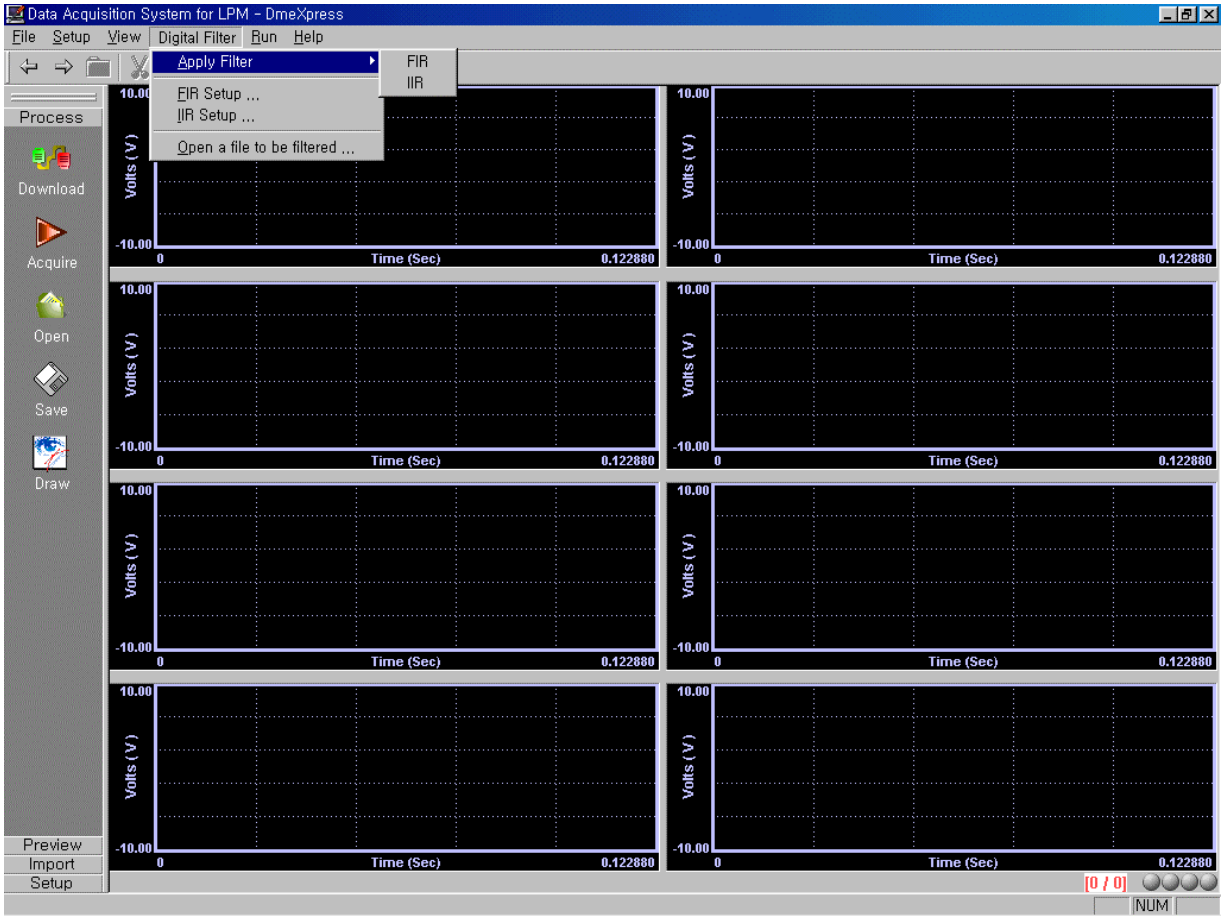


Figure 4-13. Digital Filter menu

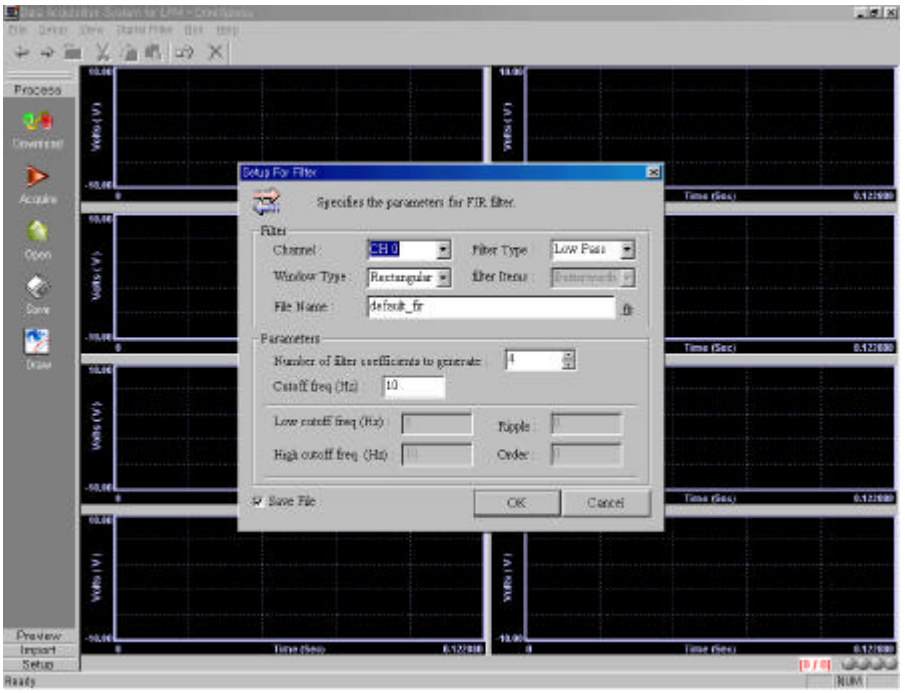


Figure 4-14. FIR Setup

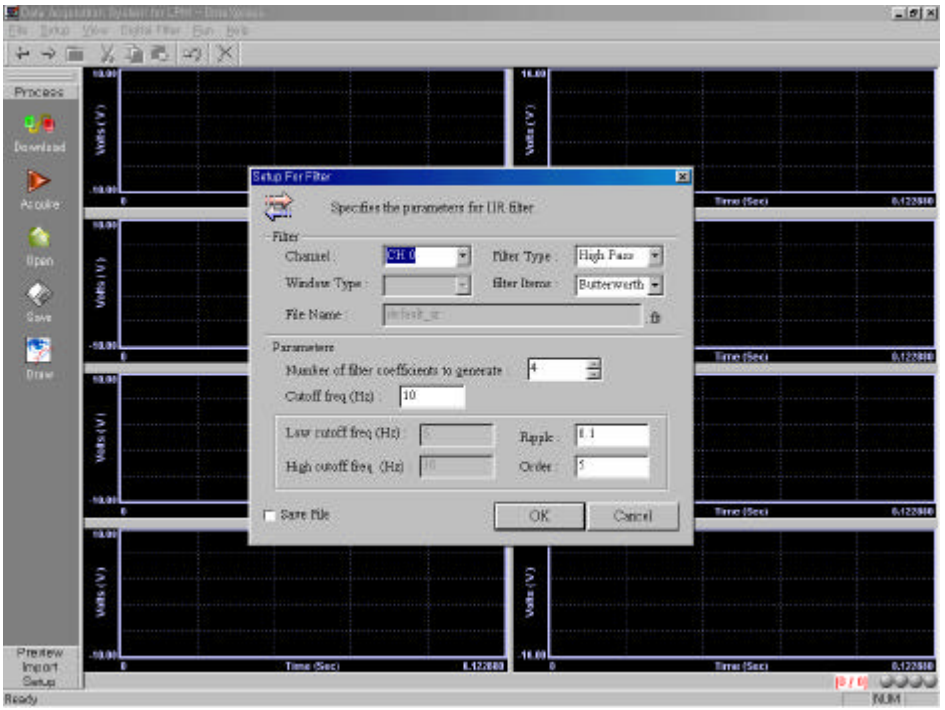


Figure 4-15. IIR Setup

4.3.4 Run

The Run menu shown in Figure 4-16 implements the actual data measurements. The type of the display window during the acquisition is dependent on the format defined in the Real Time Display form in the Overall Setup menu. The measurement results processed are stored into the pre-defined file names in the Measurement form of the Overall Setup menu for more detailed investigation of the data and for the off-line analyses later.

This menu has two submenus, namely, Acquire and LPMS Example. The Acquire menu let you start the real time signal processing for data measurements. The LPMS example let you perform an exercise with two kinds of execution files developed by KAERI. One is Location Estimation, the other Mass Estimation. The details on the algorithm and theoretical background related to the LPMS practice are described in several KAERI reports. Figure 4-17 and 4-18 illustrate the display screens corresponding to the Location Estimation and Mass Estimation, respectively.

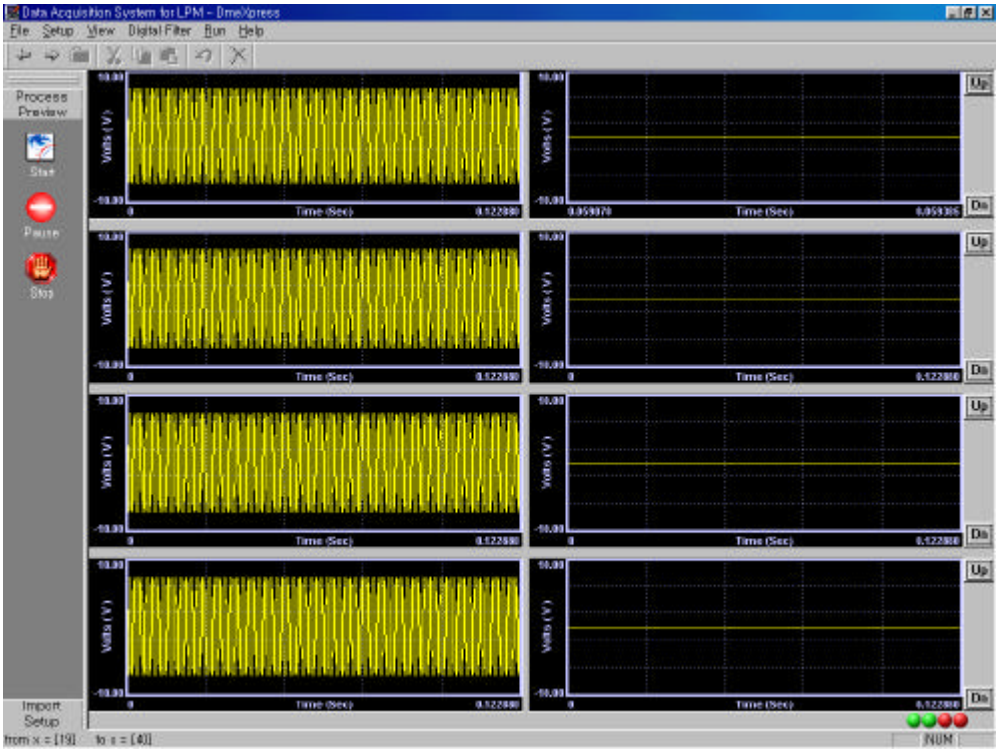


Figure 4-16 Run menu

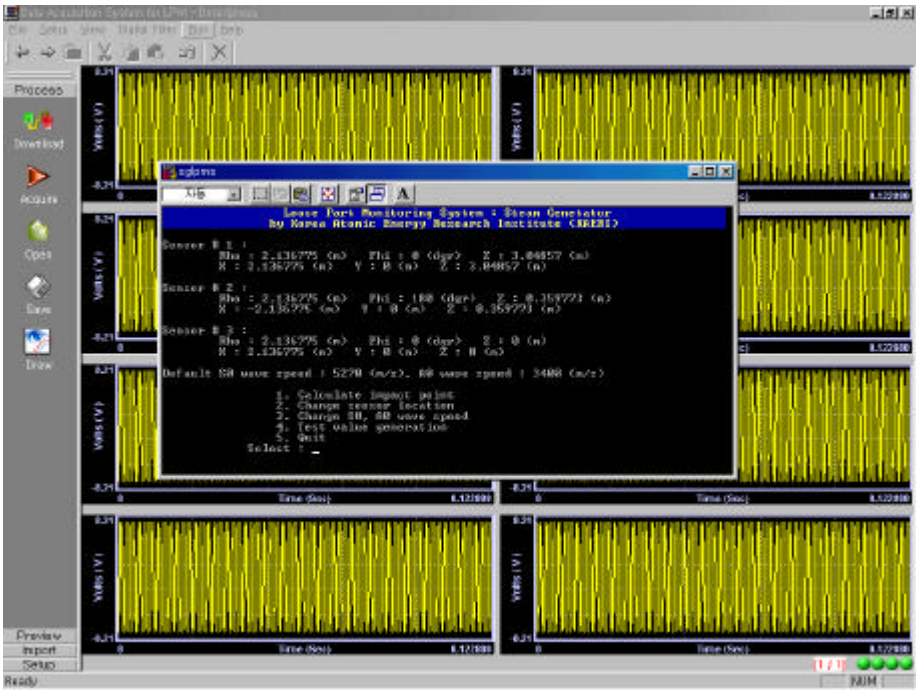


Figure 4-17. Location Estimation

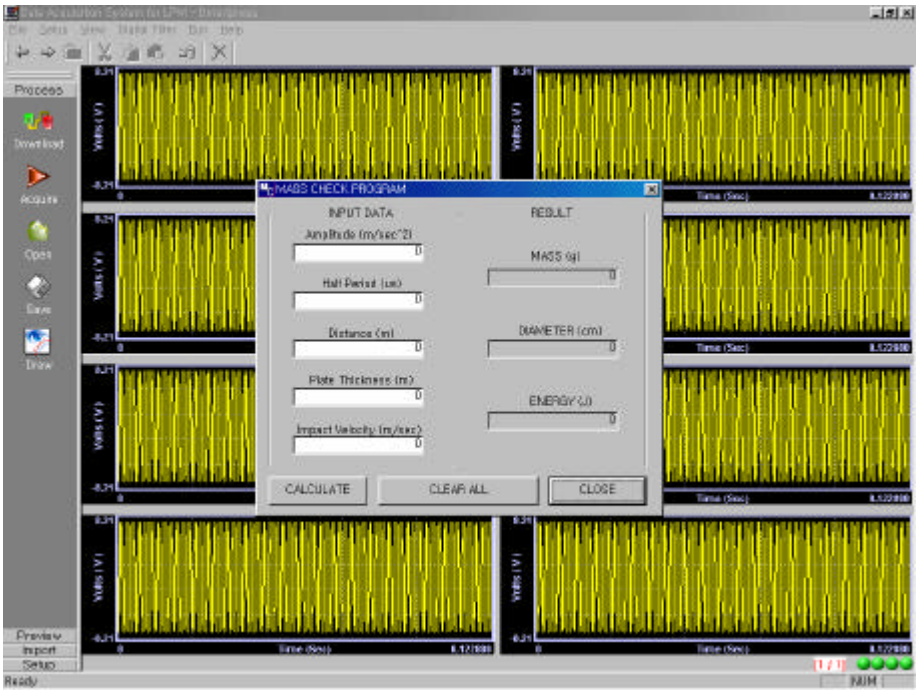


Figure 4-18. Mass Estimation

4.4 Windows Driver(VxD)

The part of VxD source code is shown in the following.

Contact to DongMyung Electronics for any questions of VxD source code.

```
//////////////////////////////////////////////////////////////////////////
//   VDASD.C -- Main code for C3x DSP Data Acquisition Device Driver
//   Use with C9 (32-bit) and MASM 6.11c from DDK
//   Copyright 2000 DongMyung Electronics Co., Ltd.
//////////////////////////////////////////////////////////////////////////

#define WANTVXDWRAPS  //  "VxD coding at C language" indicate
#include <basedef.h>
#include <vmm.h>
#include <debug.h>
……
……
                CM_Add_Empty_Log_Conf(&logconf, device, LCPRI_NORMAL,BASIC_LOG_CONF
| PRIORITY_EQUAL_LAST);
                CM_Add_Res_Des(&resource, logconf, ResType_IRQ,          &irq1,
sizeof(irq1), 0);
……
……
                case CARD_INFO:
                        CM_CallBack_Enumerator(OnEnumerate, 0);
                        plai1 = (PLAI)p ->lpvInBuffer;
                        LAI.ladi_dwMemBase[3] = plai1->ladi_dwMemBase[3];
                        plai1 = (PLAI)p ->lpvOutBuffer;
                        plai1->ladi_bIRQ = LAI.ladi_bIRQ;
                        plai1->ladi_dwIRQHandle = LAI.ladi_dwIRQHandle;
```

4.5 Dynamic Linking Library (DLL)

The part of DLL Source code is shown in the following.

Contact to DongMyung Electronics for any question of DLL source code.

```
// Xpress.cpp : Defines the initialization routines for the DLL.
//

#include "stdafx.h"
#include "Xpress.h"
#include "Utils.h"
……
……
DWORD data_process(LPVOID lpVoid)
{
        unsigned int *pc54mem;
        unsigned int *pc54mem_back;
        int block_index = 0;
        f_change = 0;
        int i=0, j=0;
        DWORD cbBytesReturned;

        pci->pci_cmd = pci->pci_cmd | 0x4;
        // pci master
        pci->pci_map[0] = pci->pci_map[0] | 0x8;
        pci->pci_map[0] = 0x80000003;                                   // enable
aperture
……
……
        if ( DeviceIoControl(hCVxD, CHK_BLOCK, (LPVOID)NULL, 0,
          (LPVOID)&LAI, sizeof(LAVXDINFO), &cbBytesReturned, NULL) ) {
                if (f_change != LAI.ladi_bReserved1) {
                        if (LAI.ladi_bReserved1 == 1) {
```

4.6 Library Reference

. VxD Function

int WINAPI open_vxd();

void WINAPI close_vxd();

int WINAPI get_vxd_buff_pos();

. PCI Function

int WINAPI pci_init();

int WINAPI pci_close();

int WINAPI put2_mail(int dat, int i);

int WINAPI get_mail(int box);

int WINAPI download(char* filename);

int WINAPI send_cmd(unsigned int ui);

. Hardware Function

int WINAPI reset();

int WINAPI install_interrupt();

int WINAPI uninstall_interrupt();

int WINAPI install_timer();

int WINAPI reset_timer();

int WINAPI set_time_count(int time_count);

int WINAPI set_gain_ctrl(int channel, int value);

double* WINAPI get_time_data_addr(int channel);

double* WINAPI get_spectral_data_addr(int channel);

int WINAPI begin_daq();

int WINAPI end_daq();

int WINAPI get_cvt(float* data, int size=MAX_CHAN);

double WINAPI get_sts_channel(int channel);

. ETC Function

void WINAPI set_hwnd(HWND hWnd);

void WINAPI begin_process();

void WINAPI close_process();

int WINAPI save_file(char *filename, int channel);

int WINAPI begin_daq()

[Description] Start data acquisition

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] If (begin_daq( ) == PASS)
printf(“ start data acquisition OK ! ”);
else printf(“start data acquisition NOK ! ”);

void WINAPI begin_process()

[Description] Begin thread process to get data of vxd buffer

[Argument] None

[Return Value] None

[Usage] begin_process( );

/* start thread process */

void WINAPI close_process()

[Description] Close thread process to get data of vxd buffer

[Argument] None

[Return Value] None

[Usage]
```
begin_process( );
/* start thread process */

close_process( );
/* finish thread process */
```

void WINAPI close_vxd()

[Description] Close vxd which is opened

[Argument] None

[Return Value] None

[Usage]
```
If(open_vxd( ) == PASS )
{
    printf(“ open vxd  OK ! ”);
    close_vxd( );
}
else printf(“open vxd NOK ! ”);
```

int WINAPI download(char* filename)

[Description] Application program downloads firmware program to memory of DSP board
If you don't want to download firmware, write hex code to ROM.
For details ROM write, contact to DongMyung electronics.

[Argument] Filename : "tst.HEX"
Afterward UpVersion File is provided at Web (www.dmelec.com)

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(download( "tst.HEX" ) == PASS )
    printf(" Down load  OK ! ");
else printf("Down load NOK ! ");
```

int WINAPI end_daq()

[Description] Finish data acquisition

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] If (end_daq( ) == PASS)
printf(“ finish data acquisition OK ! ”);
else printf(“finish data acquisition NOK ! ”);

int WINAPI get_cvt(float* data, int size)

[Description] Get current value table

[Argument] data : adc data, pointer float
Size : maximum data size
support 0 - 256, but now fixed 256

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] Float *dp;

```
status = get_cvt( dp, 256);
If(status == PASS )
    printf(" getting cvt is OK ! ");
else printf("getting cvt is NOK ! ");
```

int WINAPI get_mail(int box)

[Description] Application program gets message(mail) from pci controller of DSP board

[Argument] box : pci controller mail box
support 0 - 3, but now fixed 2

[Return Value] Mail value
(status value from dsp board)

[Usage]

```
Int status;

status = get_mail(2);
If(status == 0x55 )
    printf(" status is OK ! ");
else printf("status is NOK ! ");
```

double* WINAPI get_spectral_data_addr(int channel)

[Description] Get start address pointer of calculated spectral data block

[Argument] channel : adc channel number, 0~3

[Return Value] start address point
Pointing value of double

[Usage]
```
Double *dp;

dp = get_spectral_data_addr(2);
If(dp == NULL )
    printf(" data is non ! ");
else printf("data is available ! ");
```

double WINAPI get_sts_channel(int channel)

[Description] Get A/D value of status(trigger) channel

[Argument] channel : adc channel number, 0~3

[Return Value] Double value

[Usage] Double dp;

```
dp = get_sts_channel(2);
If(dp > TRIGGER_VALUE )
    printf(“ trigger occur ! ”);
else printf(“trigger not occur ! ”);
```

double* WINAPI get_time_data_addr(int channel)

[Description] Get start address pointer of acquisitioned time data block

[Argument] channel : adc channel number, 0~3

[Return Value] start address point
Pointing value of double

[Usage] Double *dp;

```
dp = get_time_data_addr(2);
If(dp == NULL )
    printf(" data is non ! ");
else printf("data is available ! ");
```

int WINAPI get_vxd_buff_pos()

[Description] Get current position of vxd buffer
For last acquisition data block

[Argument] None

[Return Value] address point
Integer value

[Usage]

```
int bp;

While{
/* data acquisition */
}

/* last data acquisition completed */
bp = get_vxd_buff_pos( );

/* get last data block */
```

int WINAPI install_interrupt()

[Description] Vxd installs interrupt function

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(install_interrupt( ) == PASS )
    printf(“ install interrupt OK ! ”);
else printf(“install interrupt NOK ! ”);
```

int WINAPI install_timer()

[Description] Vxd installs timer function

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(install_timer( ) == PASS )
    printf(“ install timer OK ! ”);
else printf(“install timer NOK ! ”);
```

int WINAPI open_vxd()

[Description] Open vxd to communicate with Vc3xD.vxd

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(open_vxd( ) == PASS )
    printf(“ open vxd  OK ! ”);
else printf(“open vxd NOK ! ”);
```

int WINAPI pci_close()

[Description] Close PCI chip set of DSP board

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(pci_close( ) == PASS )
    printf(“ pci close  OK ! ”);
else printf(“pci close NOK ! ”);
```

int WINAPI pci_init()

[Description] initialize PCI chip set of DSP board

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(pci_init( ) == PASS )
    printf(“ pci init  OK ! ”);
else printf(“pci init NOK ! ”);
```

int WINAPI put2_mail(int dat, int i)

[Description] application program sends message(mail) to PCI controller of DSP board

[Argument] dat : sending data, 32bit data
: pci controller mail box
support 0 - 3, but now fixed 0

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If(put2_mail(0x5555, 0 ) == PASS )
    printf(" put mail  OK ! ");
else printf("put mail NOK ! ");
```

int WINAPI reset()

[Description] Initialize hardware and software of DSP board

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] If(reset( ) == PASS) printf(“ Initialize OK ! ”);
else printf(“ Initialize NOK ! ”);

int WINAPI reset_timer()

[Description] Initialize hardware and software of DSP board vxd reset timer function

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] If(reset_timer( ) == PASS) printf(“ reset_timer OK ! ”);
else printf(“reset_timer NOK ! ”);

int WINAPI save_file(char *filename, int channel)

[Description] Save vxd buffer to file

[Argument] Filename : save filename
Channel : ADC channel, 0~3

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] If(save_file("tmp.dat",2) == PASS)
printf(" file save OK ! ");
else printf("file save NOK ! ");

int WINAPI send_cmd(unsigned int ui)

[Description] Application program sends command to DSP board
for kinds of argument, refer to chapter 3.1 table.

[Argument] Ui : command, 32bit

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If( send_cmd( ) == PASS )
    printf(“ send command OK ! ”);
else printf(“send command NOK ! ”);
```

int WINAPI set_gain_ctrl(int channel, int value)

[Description] Set gain value of amplifier in analog input of DSP board

[Argument] Channel : adc channel, 0~3
Value : amplifier value
(minimum) :
(maximum) :

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If( set_gain_ctrl( ) == PASS )
    printf(“ set gain control OK ! ”);
else printf(“set gain control NOK ! ”);
```

int WINAPI set_time_count(int time_count)

[Description] Set sampling time of ad chip in analog input of DSP board

[Argument] Time_count : sampling speed
Minimum value :
Maximum value :

[Return Value] If Pass, return PASS
Else return FAIL

[Usage]
```
If( set_time_count( ) == PASS )
    printf(“ set time count OK ! ”);
else printf(“set time count NOK ! ”);
```

void WINAPI set_hwnd(HWND hWnd)

[Description] Put message from VxD to application program for VxD interrupt

[Argument] Windows ID

[Return Value] None

[Usage]
```
If ( set_hwnd (hWnd) != 0 )
    printf(“ vxd error ! ”);
else printf(“ vxd ok ! ”);
```

int WINAPI uninstall_interrupt()

[Description] VxD uninstalls interrupt function

[Argument] None

[Return Value] If Pass, return PASS
Else return FAIL

[Usage] If(uninstall_interrupt( ) == PASS)
printf(“ uninstall interrupt OK ! ”);
else printf(“uninstall interrupt NOK ! ”);

Appendix

This chapter contents include

A. Firmware Source code (Main Program …)

B. Firmware Source code (FFT)

C. Software Source code

D. Reference Documents

E. Index of Firmware Library

F. Index of Software Library

A. Firmware Source code (Main Program ..)

(1) M31boot.asm

```
*******************************************************************************
* This File Contains Kernel program
*
*                   1. Archives :
*                   2. Filename : M31BOOT.ASM
*                   3. Version  : 1.0
*                   4. Status   :
*                   5. AUTHOR   : Yong Do Lee
*                   6. (C)      : Copyright 2001. DME. All rights reserved.
*
*                   Change History :
*                   Version         Date            AUTHORS         COMMENT
*                   1.0             APR-02-04       Y.D. Lee        original created
*
*******************************************************************************

      .sect "kernel"
************ variable ***********
                    .def            ret_dat
                    .def            mail0
                    .def            mail2
                    .def            m_stat
                    .def            f_interrupt_get ; interrupt flag values  INT0
                    .def            f_interrupt_ad      ; interrupt flag values  INT2
                    .def            f_fifo_sw           ;  ping/pong selected
                    .def            f_fifo_ptr          ; ad fifo address point
                    .def            f_mem_size          ; Mem size
                    .def            f_mem_addr0         ; Mem Address point
                    .def            f_mem_addr1         ; Mem Address point
                    .def            f_mem_addr2         ; Mem Address point
                    .def            f_mem_addr3         ; Mem Address point
                    .def            f_mem_ptr           ; Mem address point
                    .def            memaddr0            ; Mem Address ch0
                    .def            memaddr1            ; Mem Address ch1
                    .def            memaddr2            ; Mem Address ch2
                    .def            memaddr3            ; Mem Address ch3
```

```
                .def        fftaddr0        ; Mem  fft data  Address point
                .def        fftaddr1        ; Mem  fft data  Address point
                .def        fftaddr2        ; Mem  fft data  Address point
                .def        fftaddr3        ; Mem  fft data  Address point
                .def        fftmemaddr0     ; Mem fft data  Address ch0
                .def        fftmemaddr1     ; Mem fft data  Address ch1
                .def        fftmemaddr2     ; Mem fft data  Address ch2
                .def        fftmemaddr3     ; Mem fft data  Address ch3
                .def        rawaddr0        ; Mem fft data  Address ch0
                .def        rawaddr1        ; Mem fft data  Address ch1
                .def        rawaddr2        ; Mem fft data  Address ch2
                .def        rawaddr3        ; Mem fft data  Address ch3
                .def        rawmemaddr0     ; Mem fft data  Address ch0
                .def        rawmemaddr1     ; Mem fft data  Address ch1
                .def        rawmemaddr2     ; Mem fft data  Address ch2
                .def        rawmemaddr3     ; Mem fft data  Address ch3
                .def        f_timer_cnt

; ********** function **********
                .def        put_mail
                .def        get4_mail
                .def        ad_mem_move
                .def        _mail_isr
                .def        _receive_isr

                br          start

_mystack        .usect      "mystack", 50                   ; reserve 500 locations for
extern          .word       0010000H
;extern         .word       00809FC1H
stacka          .word       _mystack        ; address of mystack section
addr_first      .word       0h
down_first      .word       0h
mail0           .word       07f0030H
mail2           .word       07f0032H
m_stat          .word       07f0036H
num             .word       000000000H
num0            .word       000000000H
num1            .word       000000001H
num2            .word       000000002H
num3            .word       000000003H
num7            .word       000000007H
```

```
num9            .word       000000009H
num15           .word       00000000fH
num23           .word       000000017H
f_get_cnt       .word       000000000H
f_write         .word       00000000fH
f_read          .word       000000f00H
f_interrupt_get .word       000000000H
size            .word       000000000H
size_cnt        .word       000000000H
addr            .word       000000000H
mask            .word       000000005h
mask_e          .word       000000005h
ret_dat         .word       000000000H
msg_ack         .word       0000000eeH
f_timer_cnt     .word       000000000H
tmp_addr        .word       012000H

;******* ad variable define
f_interrupt_ad  .word       000000000H      ; ad input flag
f_fifo_sw       .word       000000000H      ;  ping/pong selected

f_fifo_size     .word       000000200H      ; ad fifo address
;f_fifo_addr    .word       000140000H      ; a/d(s->p) Data Address ( PLD )
;f_fifo_addr    .word       000310000H      ; a/d(s->p) Data Address ( PLD )
f_fifo_ptr      .word       000000000H      ; ad fifo address point

f_mem_size      .word       00000a000H      ; Mem size 40K
f_mem_ptr       .word       000000000H      ; Mem address point

f_mem_addr0     .word       memaddr0        ; Mem Address
f_mem_addr1     .word       memaddr1        ; Mem Address
f_mem_addr2     .word       memaddr2        ; Mem Address
f_mem_addr3     .word       memaddr3        ; Mem Address

fftaddr0        .word       fftmemaddr0     ; Mem Address
fftaddr1        .word       fftmemaddr1     ; Mem Address
fftaddr2        .word       fftmemaddr2     ; Mem Address
fftaddr3        .word       fftmemaddr3     ; Mem Address

rawaddr0        .word       rawmemaddr0     ; Mem Address
rawaddr1        .word       rawmemaddr1     ; Mem Address
rawaddr2        .word       rawmemaddr2     ; Mem Address
rawaddr3        .word       rawmemaddr3     ; Mem Address
```

```
*******************************************************************************
* Interrupt sevice routinue( INT 0 )
* Host --> Target command receive interrupt
* Argment : None
* Return  : Command( ret_dat, interrupt flag )
*******************************************************************************
_mail_isr:
                PUSH        ST ;Save status register
                PUSH        DP ;Save data page pointer
                PUSH        IE ;Save interrupt enable register
                PUSH        IF
                PUSHF       R0 ;upper 32 bits of R0
                PUSH        R0 ;Save lower 32 bits and
                push        ar0
                push        R7
                push        R0
                nop

get_mail
                LDI         @mail0,AR0
                ldi         *AR0, r7        ;load
                sti         r7, @ret_dat
                nop

mail_exit
                ldi         @num1,r0
                sti         r0, @f_interrupt_get
                nop

                pop         r0
                pop         r7
                pop         ar0
                POP         R0 ;lower 32 bits of R0
                POPF        R0 ;Restore upper 32 bits and
                POP         IF
                POP         IE ;Restore interrupt enable register
                POP         DP ;Restore data page register
                POP         ST ;Restore status register
                reti
```

```
*****************************************************************************
* Interrupt sevice routinue( INT 2 )
* A/D Chip half interrupt
* Argment : None
* Return  : Command( interrupt flag )
*****************************************************************************
_receive_isr:
                PUSH    ST ;Save status register
                PUSH    DP ;Save data page pointer
                PUSH    IE ;Save interrupt enable register
                PUSH    IF
                PUSHF   R0 ;upper 32 bits of R0
                PUSH    R0 ;Save lower 32 bits and
                push    ar0
                push    R7
                push    R6
                nop

get_ad
                ldi     @f_fifo_sw, r7
                not     r7, r6
                sti     r7, @f_fifo_sw          ; memory write
                nop
ad_exit
                ldi     @num1,r0
                sti     r0, @f_interrupt_ad
                nop

                pop     r6
                pop     r7
                pop     ar0
                POP     R0 ;lower 32 bits of R0
                POPF    R0 ;Restore upper 32 bits and
                POP     IF
                POP     IE ;Restore interrupt enable register
                POP     DP ;Restore data page register
                POP     ST ;Restore status register
                reti
```

```
*******************************************************************************
* Main program
*******************************************************************************
start:
;****** main init  ********
                ldp             extern                          ;deb ldp 0, DP
                nop
                ldi             @stacka, SP
                nop

;****** interrupt init ********
                ldi             @mask, r0
                and             r0, IF
                nop
                ldi             @mask_e, IE
                nop
                OR              2000h, ST
                nop
;****** interrupt init end ************
mail_down:
                call            get4_mail
                ldi             @ret_dat, r4
                sti             r4, @size
                nop
                ldi             @size, r4
                ldi             @num0, r2
                cmpi            r4, r2
                beq             mail_down_end
                call            get4_mail
                ldi             @ret_dat, r0
                sti             r0, @addr
                nop

                ldi             @down_first, r1
                ldi             @num0, r2
                cmpi            r1, r2
                bne             non_first
                sti             r0, @addr_first
                nop
                ldi             @num1, r2
                sti             r2, @down_first
                nop
```

```
non_first

                ldi             @num0, r0
                sti             r0, @size_cnt
rpt_block
                ldi             @size_cnt, r0
                ldi             @size, r2
                cmpi            r0, r2
                beq             mail_down
                call            get4_mail
                ldi             @addr,ar1       ;Destination address
                ldi             @ret_dat, r0
                sti             r0,*ar1++       ;Store the last number
                nop
                sti             ar1, @addr
                nop
                ldi             @size_cnt, r0
                ldi             @num1, r1
                addi            r1, r0
                sti             r0, @size_cnt
                br              rpt_block
mail_down_end
                nop
                ldi             @msg_ack, r5
                sti             r5, @ret_dat
                nop
                call            put_mail

;*************** exit ***************
tmp_exit:
                ldi             @addr_first, r0
                bu              r0
```

```
*****************************************************************************
* Interrupt occurs check ( INT 2 )
* Argment : None
* Return  : Command Data
*****************************************************************************
get4_mail:
                ldi             @f_interrupt_get, r0
                ldi             @num1,r7
                cmpi            r0,r7
                bne             get4_mail
                ldi             @num0,r0
                sti             r0, @f_interrupt_get
                nop
                rets

*****************************************************************************
* Interrupt occurs check ( INT 2 )
* Argment : Command Data
* Return  : None
*****************************************************************************
put_mail
                LDI             @mail2,AR0
                ldi             @ret_dat,r7
                STI             r7,*AR0         ;Store data into mail0
                nop
wait_put
                LDI             @m_stat,AR0
                ldi             *AR0, r7        ;load
                LDI             @f_read,r0
                and             r0,r7
                LDI             @f_read,r0
                cmpi            r0,r7
                beq             wait_put
                call            get4_mail       ; wait for sync
                nop
                rets
```

```
*******************************************************************************
* Interrupt occurs check ( INT 2 )
* Argment : None
* Return  : Interrupt Flag
*******************************************************************************
ad_mem_move:
                ldi             @f_interrupt_ad, r0
                ldi             @num1,r7
                cmpi            r0,r7
                bne             ad_mem_move
                ldi             @num0,r0
                sti             r0, @f_interrupt_ad
                nop
                rets


                .sect           "mem_size"
memaddr0        .usect          "memsize0", 8*1024              ; ch0 memory raw data size
memaddr1        .usect          "memsize1", 8*1024              ; ch1 memory  raw data size
memaddr2        .usect          "memsize2", 8*1024              ; ch2 memory  raw data size
memaddr3        .usect          "memsize3", 8*1024              ; ch3 memory  raw data size

fftmemaddr0     .usect          "fftdata0", 8*1024  ; ch1 memory  fft data size
fftmemaddr1     .usect          "fftdata1", 8*1024  ; ch1 memory  fft data size
fftmemaddr2     .usect          "fftdata2", 8*1024  ; ch2 memory  fft data size
fftmemaddr3     .usect          "fftdata3", 8*1024  ; ch3 memory  fft data size

rawmemaddr0     .usect          "rawdata0", 512*1024            ; ch1 memory  fft data size
rawmemaddr1     .usect          "rawdata1", 512*1024            ; ch1 memory  fft data size
rawmemaddr2     .usect          "rawdata2", 512*1024            ; ch2 memory  fft data size
rawmemaddr3     .usect          "rawdata3", 512*1024            ; ch3 memory  fft data size

                .sect           "vectors"
INT0            br              _mail_isr       ; 0x809FC1 0x001
INT1            br              $                               ; 0x809FC1 0x002
INT2            br              _receive_isr    ; 0x809FC1 0x004


                .end
```

(2) Sub.asm

```
*******************************************************************************
* This File Contains Subroutine
*
*                   1. Archives :
*                   2. Filename : SUB.ASM
*                   3. Version  : 1.0
*                   4. Status   :
*                   5. AUTHOR   : Yong Do Lee
*                   6. (C)      : Copyright 2001. DME. All rights reserved.
*
*                   Change History :
*                   Version         Date            AUTHORS         COMMENT
*                   1.0             APR-02-04       Y.D. Lee        original created
*
*******************************************************************************

; variable
                    .ref            ret_dat

                    .ref            f_interrupt_get ; interrupt flag values  INT0
                    .ref            f_interrupt_ad      ; interrupt flag values  INT2

;                   .ref            f_fifo_size         ; ad fifo address
;                   .ref            f_fifo_addr         ; a/d(s->p) Data Address ( PLD )
                    .ref            f_fifo_ptr          ; ad fifo address point

                    .ref            f_mem_size          ; Mem size

                    .ref            f_mem_addr0         ; Mem Address
                    .ref            f_mem_addr1         ; Mem Address
                    .ref            f_mem_addr2         ; Mem Address
                    .ref            f_mem_addr3         ; Mem Address
                    .ref            f_mem_ptr           ; Mem address point

                    .ref            memaddr0            ; Mem Address ch0
                    .ref            memaddr1            ; Mem Address ch1
                    .ref            memaddr2            ; Mem Address ch2
                    .ref            memaddr3            ; Mem Address ch3

                    .ref            rawaddr0            ; raw Mem Address ch0
                    .ref            rawaddr1            ; raw Mem Address ch1
                    .ref            rawaddr2            ; raw Mem Address ch2
                    .ref            rawaddr3            ; raw Mem Address ch3
```

```
                .ref        rawmemaddr0     ; raw Mem Address ch0
                .ref        rawmemaddr1     ; raw Mem Address ch1
                .ref        rawmemaddr2     ; raw Mem Address ch2
                .ref        rawmemaddr3     ; raw Mem Address ch3

                .ref        mail0
                .ref        mail2                       ; pci address
                .ref        m_stat

                .ref        put_mail
                .ref        get4_mail
                .ref        ad_mem_move

; function
fp              .set        AR3
                .global _cput_mail
                .global _cget4_mail
                .global     _cget4_mail_flag
                .global _cget_ad
                .global     _cget_ad_ch0
                .global     _cget_ad_ch1
                .global     _cget_ad_ch2
                .global     _cget_ad_ch3
                .global _cget_ad_flag
                .global _cad_memory_reset
                .global     _cad_init
                .global     _cad_start
                .global     _cad_end
                .global _gain_ctrl_ch0
                .global     _gain_ctrl_ch1
                .global _gain_ctrl_ch2
                .global     _gain_ctrl_ch3
                .global _cad_fifo_reset                 ; ad fifo reset
                .global _cfifo_garbage_clear

                .global     _ctest_addr_read    ;deb

                .global     _c3xtimer           ; timer function
                .global _c3xtimer_reset     ; timer reset
                .global _c3xtimer_init                  ; period setup H1 Clock Control

    .text
; common variable
tst_dat01       .word 012345671H
num0                    .word 000000000H
num1                    .word 000000001H
f_read                  .word 000000f00H
```

```
; address defines *********************************
ad_start_addr       .word 000610010H ; ad start
gain_ch0_addr       .word 000610020H ; amp gain ch 0
gain_ch1_addr       .word 000610021H ; amp gain ch 1
gain_ch2_addr       .word 000610022H ; amp gain ch 2
gain_ch3_addr       .word 000610023H ; amp gain ch 3
ad_fifo_addr        .word 0006100ffH  ; ad fifo reset
f_fifo_addr         .word 000600000H ; a/d(s->p) Data Address ( PLD )
; address defines end*****************************

;f_mem_addr_start0                  .word memaddr0    ; Mem Address
;f_mem_addr_start1                  .word memaddr1    ; Mem Address
;f_mem_addr_start2                  .word memaddr2    ; Mem Address
;f_mem_addr_start3                  .word memaddr3    ; Mem Address

;f_mem_addr_restart0                .word memaddr0    ; Mem Address
;f_mem_addr_restart1                .word memaddr1    ; Mem Address
;f_mem_addr_restart2                .word memaddr2    ; Mem Address
;f_mem_addr_restart3                .word memaddr3    ; Mem Address

f_rawtomove_ch0   .word memaddr0    ; Mem Address
f_rawtomove_ch1   .word memaddr1    ; Mem Address
f_rawtomove_ch2   .word memaddr2    ; Mem Address
f_rawtomove_ch3   .word memaddr3    ; Mem Address

f_mem_addr_start0 .word rawmemaddr0                 ; Mem Address
f_mem_addr_start1 .word rawmemaddr1                 ; Mem Address
f_mem_addr_start2 .word rawmemaddr2                 ; Mem Address
f_mem_addr_start3 .word rawmemaddr3                 ; Mem Address

f_mem_addr_restart0                .word rawmemaddr0                ; Mem Address
f_mem_addr_restart1                .word rawmemaddr1                ; Mem Address
f_mem_addr_restart2                .word rawmemaddr2                ; Mem Address
f_mem_addr_restart3                .word rawmemaddr3                ; Mem Address


f_test_addr         .word 00124000H

********************  Timer Variable ***************************************
t0gcr                   .word           000808030h ;  Timer 1 gcr address
t0cr                    .word           000808034h ;  Timer 1 tc address
t0pr                    .word           000808038h ;  Timer 1 tp address

t0prval     .word       000000128h ;  Timer 1 period value
timecr                  .word           0000002C1h ;  Timer global control register
                                       ;0010 1100 0001
timest                  .word           000000301h ;  Timer reset and run mask
timehd                  .word           0FFFFFF3Fh ;  Timer hold mask
timect                  .word           000000080h ;  Timer continue mask
********************  Timer Variable ***************************************
```

```
*****************************************************************************
* Host --> Target Command Receive( INT0 assign )
* Argment : None
* Return  : Command( ret_dat )
* notice    : unlimit wating until interrupt
*****************************************************************************
_cget4_mail:
                call            get4_mail
                ldiu            @ret_dat, r0                    ;deb
                rets

*****************************************************************************
* Host --> Target Interrupt Flag Check
* Argment : None
* Return  : Interrupt Flag
*                   0 : no interrupt
*                   1 : interrupt occurs
*****************************************************************************
_cget4_mail_flag:
                ldiu            @f_interrupt_get, r0                    ;deb
                rets

*****************************************************************************
* Timer 1 Install
* Argment : None
* Return  : None
*****************************************************************************
_c3xtimer:                      ; Timer 0
                PUSH            AR0
                PUSH            AR1
                PUSH            AR2

                LDI             @t0gcr,AR0      ; Load Timer 0 GCR Address in AR0
                LDI             @t0cr,AR1       ; Load Timer 0 Ctr Reg Addr in AR1
                LDI             @t0pr,AR2       ; Load Timer 0 Per Reg Addr in AR2
                LDI             @timehd,R2      ; Load Timer hold mask
                LDI             0,R5            ; Clear R4
                STI             R5,*AR0         ; Clear Timer 0 GCR
                STI             R5,*AR1         ; Clear Timer 0 Counter Register
                LDI             @t0prval,R1     ; Load Timer 0 period value in R1
                STI             R1,*AR2         ; Set Timer 0 period value (1k)
                LDI             @timecr,R1      ; Load GCR value in R1
                STI             R1,*AR0         ; Set Timer 0 Global Control Reg

                POP             AR2
                POP             AR1
                POP             AR0
                RETSU                           ; Return
```

```
*******************************************************************************
* Timer 1 Stop
* Argment : None
* Return  : None
*******************************************************************************
_c3xtimer_reset:
                PUSH    AR0
                PUSH    AR1
                PUSH    AR2

                LDI     @t0gcr,AR0      ; Load Timer 0 GCR Address in AR0
                LDI     @t0cr,AR1       ; Load Timer 0 Ctr Reg Addr in AR1
                LDI     @t0pr,AR2       ; Load Timer 0 Per Reg Addr in AR2
                LDI     @timehd,R2      ; Load Timer hold mask
                LDI     0,R5            ; Clear R4
                STI     R5,*AR0         ; Clear Timer 0 GCR
                STI     R5,*AR1         ; Clear Timer 0 Counter Register
                STI     R5,*AR2         ; Clear Timer 0 Period Register

                POP     AR2
                POP     AR1
                POP     AR0
                RETSU                                   ; Return

*******************************************************************************
* Timer 1 Setup
* Argment : Period Count
* Return  : None
*******************************************************************************
_c3xtimer_init:
                .if     .REGPARM = 0
                push    fp
                ldi     sp,fp
                ldi     *-fp(2), ar2
                .endif

                sti     ar2, @t0prval

                .if     .REGPARM = 0
                pop     fp
                .endif
                rets
```

```
*****************************************************************************
* FiFO 1K(512 * 2)  ping/pong
* FIFO --> Circular Memory
* Argment : None
* Return  : None
*****************************************************************************
_cget_ad:
                push        ar0
                push        ar1
                push        ar2
                push        ar3
                push        ar4

                call        ad_mem_move         ; flag check & reset

                LDI         @f_fifo_addr, AR0
                ldi         @rawaddr0, ar1      ; ch0 point
                ldi         @rawaddr1, ar2      ; ch1 point
                ldi         @rawaddr2, ar3      ; ch2 point
                ldi         @rawaddr3, ar4      ; ch3 point
                nop

                ldi         (512/4) - 1, rc     ; # attention  : fifo size / use ch.
                rptb        endloop-1
startloop
                ldi         *AR0, R1            ; ch0 data move
                sti         R1, *AR1++
                ldi         *AR0, R1            ; ch1 data move
                sti         R1, *AR2++
                ldi         *AR0, R1            ; ch2 data move
                sti         R1, *AR3++
                ldi         *AR0, R1            ; ch3 data move
                sti         R1, *AR4++
                nop
endloop
                sti         ar1, @rawaddr0      ; memory point save ch0
                sti         ar2, @rawaddr1      ; memory point save ch1
                sti         ar3, @rawaddr2      ; memory point save ch2
                sti         ar4, @rawaddr3      ; memory point save ch3
                nop
cget_ad_end
                pop         ar4
                pop         ar3
                pop         ar2
                pop         ar1
                pop         ar0
                rets
```

```
*******************************************************************************
* FIFO Reset
* Argment : None
* Return  : None
*******************************************************************************
_cad_fifo_reset:    LDI         @ad_fifo_addr, AR0
                    ldi         @num1, r7
                    STI         r7,*AR0         ;Store data into mail0
                    nop
                    rets
*******************************************************************************
* FIFO Reset( garbage read )
* Argment : None
* Return  : None
*******************************************************************************
_cfifo_garbage_clear:push       ar0
                    push        ar1
                    LDI         @f_fifo_addr, AR0
                    ldi         @f_test_addr, ar1
                    nop
                    ldi         (1024 - 1), rc      ; # attention  : fifo size / use ch.
                    rptb        clearloop - 1
                    ldi         *AR0, R1            ; ch0 data move
                    sti         R1, *AR1
                    nop
clearloop           pop         ar1
                    pop         ar0
                    rets
*******************************************************************************
* Trigger Level Check( Ch 0 )
* Argment : None
* Return  : Trigger Channel Data
*******************************************************************************
_cget_ad_ch0:       push        ar0
                    .if         .REGPARM = 0
                    push        fp
                    ldi         sp,fp
                    .endif
                    .if         .BIGMODEL
                    ldp         f_mem_addr_start0
                    .endif
                    LDI         @f_mem_addr_start0, AR0
                    LDI         *AR0++, R0
                    STI         AR0, @f_mem_addr_start0
                    nop
                    .if         .REGPARM = 0
                    pop         fp
                    .endif
                    pop         ar0
                    rets
```

```
*****************************************************************************
* Trigger Level Check( Ch 1 )
* Argment : None
* Return  : Trigger Channel Data
*****************************************************************************
_cget_ad_ch1:   push        ar0

                .if         .REGPARM = 0
                push        fp
                ldi         sp,fp
                .endif
                .if         .BIGMODEL
                ldp         f_mem_addr_start1
                .endif
                LDI         @f_mem_addr_start1, AR0
                LDI         *AR0++, R0
                STI         AR0, @f_mem_addr_start1
                nop
                .if         .REGPARM = 0
                pop         fp
                .endif

                pop         ar0
                rets

*****************************************************************************
* Trigger Level Check( Ch 2 )
* Argment : None
* Return  : Trigger Channel Data
*****************************************************************************
_cget_ad_ch2:   push        ar0

                .if         .REGPARM = 0
                push        fp
                ldi         sp,fp
                .endif
                .if         .BIGMODEL
                ldp         f_mem_addr_start2
                .endif
                LDI         @f_mem_addr_start2, AR0
                LDI         *AR0++, R0
                STI         AR0, @f_mem_addr_start2
                nop
                .if         .REGPARM = 0
                pop         fp
                .endif

                pop         ar0
                rets
```

```
*****************************************************************************
* Trigger Level Check( Ch 3 )
* Argment : None
* Return  : Trigger Channel Data
*****************************************************************************
_cget_ad_ch3:   push        ar0

                .if         .REGPARM = 0
                push        fp
                ldi         sp,fp
                .endif
                .if         .BIGMODEL
                ldp         f_mem_addr_start3
                .endif
                LDI         @f_mem_addr_start3, AR0
                LDI         *AR0++, R0
                STI         AR0, @f_mem_addr_start3
                nop
                .if         .REGPARM = 0
                pop         fp
                .endif

                pop         ar0
                rets
```

```
*****************************************************************************
* Memory Point Reset
* Argment : None
* Return  : None
*****************************************************************************
_cad_memory_reset:
                push            ar1
; ch0
                ldi             @f_mem_addr_restart0, ar1
                sti             ar1, @rawaddr0                  ; memory point save ch0
                nop
                ldi             @f_mem_addr_restart0, ar1
                sti             ar1, @f_mem_addr_start0
                nop

; ch1
                ldi             @f_mem_addr_restart1, ar1
                sti             ar1, @rawaddr1                  ; memory point save ch1
                nop
                ldi             @f_mem_addr_restart1, ar1
                sti             ar1, @f_mem_addr_start1
                nop

; ch2
                ldi             @f_mem_addr_restart2, ar1
                sti             ar1, @rawaddr2                  ; memory point save ch2
                nop
                ldi             @f_mem_addr_restart2, ar1
                sti             ar1, @f_mem_addr_start2
                nop



; ch3
                ldi             @f_mem_addr_restart3, ar1
                sti             ar1, @rawaddr3                  ; memory point save ch3
                nop
                ldi             @f_mem_addr_restart3, ar1
                sti             ar1, @f_mem_addr_start3
                nop

                pop             ar1
                rets
```

```
*****************************************************************************
* A/D half Interrupt occurs check( INT2 assign)
* Argment : None
* Return  : Interrupt flag
*****************************************************************************
_cget_ad_flag:  ldiu        @f_interrupt_ad, r0
                rets
*****************************************************************************
* FiFO Memory Reset
* Argment : None
* Return  : None
*****************************************************************************
_cad_init:      ldiu        @num1, r0       ;deb
                rets
*****************************************************************************
* A/D Chip start( Data Acquisitio begin )
* Argment : None
* Return  : None
*****************************************************************************
_cad_start:     LDI         @ad_start_addr, AR0
                ldi         @num1, r7
                STI         r7,*AR0                     ;Store data into mail0
                nop
                rets
*****************************************************************************
* A/D Chip stop( Data Acquisitio stop )
* Argment : None
* Return  : None
*****************************************************************************
_cad_end:       LDI         @ad_start_addr, AR0
                ldi         @num0, r7
                STI         r7,*AR0                     ;Store data into mail0
                nop
                rets
*****************************************************************************
* Target --> Host Command trans
* Argment : Command
* Return  : None
*****************************************************************************
_cput_mail:     .if         .REGPARM = 0
                push        fp
                ldi         sp,fp
                ldi         *-fp(2), ar2
                .endif
                sti         ar2, @ret_dat
                call        put_mail
                .if         .REGPARM = 0
                pop         fp
                .endif
                rets
```

```
*****************************************************************************
* Channel 0 gain control
* Argment : gain data
* Return  : None
*****************************************************************************
_gain_ctrl_ch0:
                .if             .REGPARM = 0
                push            fp
                ldi             sp,fp
                ldi             *-fp(2), ar2
                .endif

                LDI             @gain_ch0_addr, ar0
                sti             ar2, *ar0

                .if             .REGPARM = 0
                pop             fp
                .endif
                rets

*****************************************************************************
* Channel 1 gain control
* Argment : gain data
* Return  : None
*****************************************************************************
_gain_ctrl_ch1:
                .if             .REGPARM = 0
                push            fp
                ldi             sp,fp
                ldi             *-fp(2), ar2
                .endif

                LDI             @gain_ch1_addr, ar0
                sti             ar2, *ar0

                .if             .REGPARM = 0
                pop             fp
                .endif
                rets
```

```
*****************************************************************************
* Channel 2 gain control
* Argment : gain data
* Return  : None
*****************************************************************************
_gain_ctrl_ch2:
                .if             .REGPARM = 0
                push            fp
                ldi             sp,fp
                ldi             *-fp(2), ar2
                .endif

                LDI             @gain_ch2_addr, ar0
                sti             ar2, *ar0

                .if             .REGPARM = 0
                pop             fp
                .endif
                rets

*****************************************************************************
* Channel 3 gain control
* Argment : gain data
* Return  : None
*****************************************************************************
_gain_ctrl_ch3:
                .if             .REGPARM = 0
                push            fp
                ldi             sp,fp
                ldi             *-fp(2), ar2
                .endif

                LDI             @gain_ch3_addr, ar0
                sti             ar2, *ar0

                .if             .REGPARM = 0
                pop             fp
                .endif
                rets
```

(3) C31.c

```
////////////////////////////////////////////////////////////////////////////
// This File Contains Main Program & Window Function
//
//                  1. Archives :
//                  2. Filename : C31.c
//                  3. Version  : 1.0
//                  4. Status   :
//                  5. AUTHOR   : Yong Do Lee
//                  6. (C)      : Copyright 2001. DME. All rights reserved.
//
//                  Change History :
//                  Version         Date            AUTHORS         COMMENT
//                  1.0             APR-02-04       Y.D. Lee        original created
//
////////////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <intrin.h>  // floating point function

#include "add.h"
#include "vecs.h"
#include "values.h"
#include "math.h"

/*=========================================================================*/
/* GLOBAL CONSTANTS                                                        */
/*=========================================================================*/
#define FFT_SIZE                8192
#define HALF_FFT_SIZE           FFT_SIZE/2
#define LOG_SIZE                13
#define BITREV                  1                       //  Bit reverse will be performed

#define CH_CNT                  4
#define FIFO_SIZE               512
#define MEM_SIZE                8*1024
#define ONE_READ_SIZE           FIFO_SIZE / CH_CNT  // 1Cycle

#define CIRCULAR_SIZE           ( MEM_SIZE / (FIFO_SIZE / CH_CNT) )   // circular buffer size
#define EXIT_SIZE               ( MEM_SIZE / (FIFO_SIZE / CH_CNT) )
```

```
extern int cput_mail();
extern int cget4_mail();
extern int cget4_mail_flag();
extern int cget_ad();
extern int cget_ad_ch0();
extern int cget_ad_ch1();
extern int cget_ad_ch2();
extern int cget_ad_ch3();
extern int cget_ad_flag();
extern int cad_memory_reset();
extern int cad_init();
extern int cad_start();
extern int cad_end();
extern int cad_fifo_reset();                    // ad fifo reset
extern int cfifo_garbage_clear();

extern int gain_ctrl_ch0();
extern int gain_ctrl_ch1();
extern int gain_ctrl_ch2();
extern int gain_ctrl_ch3();

extern int ffft_rl();

extern int c3xtimer();                                          // timer 1
extern int c3xtimer_reset();    // timer reset
extern int c3xtimer_init();     // period setup(H1 Clock )

volatile int   *RAWBufferIn_CH0 = (volatile int *)   0x00200000;
volatile int   *RAWBufferIn_CH1 = (volatile int *)   0x00280000;
volatile int   *RAWBufferIn_CH2 = (volatile int *)   0x00300000;
volatile int   *RAWBufferIn_CH3 = (volatile int *)   0x00380000;

volatile int   *FFTBufferIn_CH0 = (volatile int *)   0x00110000;
volatile int   *FFTBufferIn_CH1 = (volatile int *)   0x00112000;
volatile int   *FFTBufferIn_CH2 = (volatile int *)   0x00114000;
volatile int   *FFTBufferIn_CH3 = (volatile int *)   0x00116000;
volatile int   *FFTToIEEE_CH0   = (volatile int *)   0x00128000;
volatile int   *FFTToIEEE_CH1   = (volatile int *)   0x0012A000;
volatile int   *FFTToIEEE_CH2   = (volatile int *)   0x0012C000;
volatile int   *FFTToIEEE_CH3   = (volatile int *)   0x0012E000;

volatile float *FFT_INPUT       = (volatile float *) 0x00118000;
volatile float *FFTBufferOut    = (volatile float *) 0x0011A000;
volatile float *SinTable        = (volatile float *) 0x0010E000;
```

```
////////////////////////////////////////////////////////////////////////////////
// global variable define
unsigned int        block_start=0, block_cnt=0;  // circular block count : memory size / 512
unsigned int        exit_cnt=0;                  // Exit Flag
unsigned int        chk_point=0;                 // start point check ch0
unsigned int        Auto_Mode=0;
unsigned int        Trig_Mode=0;
unsigned int        Stationary_Mode=0;
unsigned int        chk_data = 0x00003000 ;
unsigned int        chk_channel=0;
unsigned int        pre_trigger=4;
unsigned int        Trigger_Quantity_Reference=30;
unsigned int        Trigger_Quantity_Response =30;
unsigned int        Winfor_refernce=0;
unsigned int        Winfor_response=0;
unsigned int        Frame_Size=8192;
unsigned int        Average_Count =1;
unsigned int        Overlap=4;
unsigned int        Exp_Decay_Reference=100;
unsigned int        Exp_Decay_Response =100;
unsigned int        Data_Processing =4;

unsigned int        FFT_Size=0;
unsigned int        Half_FFT_Size=0;
unsigned int        Log_Size=0;
unsigned int        Bitrev=0;

unsigned int        Ch_Cnt=4;
unsigned int        Fifo_Size=0;
unsigned int        Mem_Size=0;
unsigned int        One_Read_Size=0;

unsigned int        Circular_Size=0;
unsigned int        Exit_Size=0;

unsigned int garbage_clear=0;
////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////
// Circular variable init
void circular_init()
{
                int i;

                FFT_Size       = Frame_Size * Average_Count;          // max 8K
                Half_FFT_Size  = Frame_Size/2;
                Log_Size       = 13;                    // default
                for (i=0;i<15;i++){ if( Frame_Size == (int) pow(2,i) ) {Log_Size = i;break;} }
                Bitrev         = BITREV;
                Ch_Cnt         = CH_CNT;                // 4 ch
                Fifo_Size      = FIFO_SIZE;             // 1K/2 = 512
                Mem_Size       = Frame_Size;            // 8K
                One_Read_Size  = Fifo_Size/Ch_Cnt;      // 128
                Circular_Size  = FFT_Size/(Fifo_Size/Ch_Cnt);          // 64
                Exit_Size      = FFT_Size/(Fifo_Size/Ch_Cnt);          // 64
}
////////////////////////////////////////////////////////////////////////////
// Circular Data --> Liner Data convertion( Auto/Trigger Mode )
int Data_Arrange( volatile int *RAWBufferIn, volatile int *FFTBufferIn )
{
                int i,j;
                int raw_ad_data;
                ////////////////////////////////////////////////////////////
                // Channel convert
                ////////////////////////////////////////////////////////////
                j = 0;
                for (i=block_start*(One_Read_Size);i<Circular_Size*(One_Read_Size);i++){
                                raw_ad_data = ( 0x0000FFFF & RAWBufferIn[i] );
                                if ( raw_ad_data >= 0x8000 ) {
                                                raw_ad_data ^= 0xffff;
                                                raw_ad_data  = (float)raw_ad_data * -1;

                                }
                                FFT_INPUT[j++] = ((float) raw_ad_data) * 100;
                }
                for (i=0;i<block_start*(One_Read_Size);i++) {
                                raw_ad_data = ( 0x0000FFFF & RAWBufferIn[i] );
                                if ( raw_ad_data >= 0x8000 ) {
                                                raw_ad_data ^= 0xffff;
                                                raw_ad_data  = (float)raw_ad_data * -1;
                                }
                                FFT_INPUT[j++] = ((float) raw_ad_data) * 100;
                }
                // dsp 31 data format --->  ieee data format
                for (i=0;i<Frame_Size;i++) {            FFTBufferIn[i] = (int) (FFT_INPUT[i]/100); }
                // Channel End
                ////////////////////////////////////////////////////////////
                return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////
// Circular Data --> Liner Data convertion( Stationary Mode )
int Data_Arrange_Stationary( volatile int *RAWBufferIn, volatile int *FFTBufferIn , int raw_ptr)
{
                int i,j;
                int raw_ad_data;

                ////////////////////////////////////////////////////////////
                // Channel convert
                ////////////////////////////////////////////////////////////
                j = 0;
                for (i=Frame_Size * raw_ptr; i< (Frame_Size * raw_ptr) + Frame_Size; i++) {
                                raw_ad_data = ( 0x0000FFFF & RAWBufferIn[i] );
                                if ( raw_ad_data >= 0x8000 ) {
                                                raw_ad_data ^= 0xffff;
                                                raw_ad_data  = (float)raw_ad_data * -1;

                                }
                                FFT_INPUT[j++] = ((float) raw_ad_data) * 100;
                }
                // dsp 31 data format --->  ieee data format
                for (i=0;i<Frame_Size;i++) {
                                FFTBufferIn[i] = (int) (FFT_INPUT[i]/100);
                }

                // Channel End
                ////////////////////////////////////////////////////////////

                return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////////
// Raw To FFT Converter
//                    1. integer --> float
//                    2. Window Data Converter
//                      Window Flag : 1=window reference(trigger channel), 0=window response
//                      Window : 0=time,1=time&window,2=spectra,3=time&spectra,4=avg spectra
//                    3. FFT
//                    4. float --> integer
//                    5. FFTBufferIn : Raw Data , FFTToIEEE   : Process Data( FFT )
int RawToFFT( volatile int *FFTBufferIn , volatile int *FFTToIEEE, int Window_Flag )
{
                    int i;

                    int fft_return;

                    ////////////////////////////////////////////////////////////////
                    // Channel convert
                    ////////////////////////////////////////////////////////////////

                    for (i=0;i<Frame_Size;i++) {
                                        FFT_INPUT[i] = (float) FFTBufferIn[i] * 0.000305;
                    }

                    // Data_processing
                    // 0 : time, 1 : time & window , 2 : spectra, 3 : time & spectra, 4 : avg spectra

                    if ( Data_Processing == 0 ) return 0;

                    if ( Window_Flag == 0 ) {
                                        switch( Winfor_response ) {
                                                            case 0 :            // no windows
                                                                                break;
                                                            case 1 :
                                                                                Hanning_Window( );
                                                                                break;
                                                            case 2 :
                                                                                Exponential_Window( Window_Flag );
                                                                                break;
                                                            case 3 :
                                                                                Rectangular_Window( Window_Flag );
                                                                                break;
                                                            case 4 :
                                                                                Hamming_Window( );
                                                                                break;

                                                            default :
                                                                                break;
                                        }
                     }
```

```
        else {
                switch( Winfor_refernce ) {
                        case 0 :        // no windows
                                        break;
                        case 1 :
                                        Hanning_Window( );
                                        break;
                        case 2 :
                                        Exponential_Window( Window_Flag );
                                        break;
                        case 3 :
                                        Rectangular_Window( Window_Flag );
                                        break;
                        case 4 :
                                        Hamming_Window( );
                                        break;

                        default :
                                        break;
                }

        }
        switch( Data_Processing ) {
                case 1 :                        // time & window

                        // dsp 31 data format --->  ieee data format
                        for (i=0;i<Frame_Size;i++) {
                                FFTToIEEE[i] = FFT_INPUT[i]*10000;
                        }
                        break;
                case 2 :                        // spectra
                case 3 :                        // time & spectra
                case 4 :                        // avg spectra
                        fft_return = ffft_rl( Frame_Size, Log_Size, FFT_INPUT,
                          FFTBufferOut,SinTable,Bitrev);

                        // dsp 31 data format --->  ieee data format
                        for (i=0;i<Frame_Size;i++) {
                                FFTToIEEE[i] = FFTBufferOut[i];
                        }
                        break;
                default :
                        break;
        }


        // Channel End
        ////////////////////////////////////////////////////////////

        return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////////
// The Hamming command defines Hamming window for the filter.
// Filter : Low Pass Filter
// Equation :
//                      w(n) = Window value at point n
//                      N    = Filter Length
//                      PI   = 3.14159....
//                      w(k) = 0.54 - 0.46 COS( 2*PI * ( k/(N-1)) ) ,  k= 0 ~ N-1
int Hamming_Window( )
{
                      int i;
                      float Hamming_Data;

                      for (i=0;i<Frame_Size;i++) {
                          Hamming_Data = 0.54 - 0.46 * cos( 2.0*PI*( (float)i / ( (float)Frame_Size-1.0)) );
                          FFT_INPUT[i] = FFT_INPUT[i] * Hamming_Data;
                      }
}


////////////////////////////////////////////////////////////////////////////////
// The Hanning command defines Hanning window for the filter.
// Filter : Low Pass Filter
// Equation
//                      w(n) = Window value at point n
//                      N    = Filter Length
//                      PI   = 3.14159....
//                      w(k) = 0.5 ( 1 - COS( 2*PI * ( k/(N-1)) ) ,  k= 0 ~ N-1
int Hanning_Window( )
{
                      int i;
                      float Hanning_Data;

                      for (i=0;i<Frame_Size;i++) {
                          Hanning_Data = 0.5 * ( 1.0 - cos( 2.0*PI*( (float)i / ( (float)Frame_Size-1.0)) ));
                          FFT_INPUT[i] = FFT_INPUT[i] * Hanning_Data;
                      }
}
```

```
////////////////////////////////////////////////////////////////////////////
// You Must enter a percentage in the Decay Rate % field. The number used is a percent of
// the ordinate range and didcatates the percent of signal drop off at the frame midpoint.
// Equation :        x(n) * exp(-#*n)
//                                    n    = Exponential Decay
//                                    #    = -(2/N) * log( n / 100 )      ; #- define unlimit
//                                    N    = Filter Length
int Exponential_Window( int Window_Flag)
{

                    float alpha;
                    float Exponential_Data;
                    unsigned int Exponential_Decay;
                    int i;


                    if ( Window_Flag ) Exponential_Decay = Exp_Decay_Reference;
                    else Exponential_Decay = Exp_Decay_Response;

                    for (i=0;i<Frame_Size;i++) {
                        alpha =  -((2.0/ (float)Frame_Size ) * log( (float)Exponential_Decay/100.0));
                        Exponential_Data = FFT_INPUT[i] * exp( (-alpha) * (float)i );
                        FFT_INPUT[i] = Exponential_Data;
                    }
}
////////////////////////////////////////////////////////////////////////////
int Rectangular_Window( int Window_Flag )
{

                    int i;
                    int Data_Quantity;

                    Data_Quantity=0;

                    if ( Window_Flag ) Data_Quantity = ( Frame_Size / 100 ) * Trigger_Quantity_Reference;
                    else Data_Quantity = ( Frame_Size / 100 ) * Trigger_Quantity_Response;

                    ////////////////////////////////////////////////////////////////
                    // Channel convert
                    ////////////////////////////////////////////////////////////////
                    for (i=0;i<Frame_Size;i++) {
                        FFT_INPUT[i] = FFT_INPUT[i] * 100;
                    }

                    // data move
                    for (i=0;i<Frame_Size;i++) {
                        if( (i <= Data_Quantity ) ) { FFT_INPUT[i] = (FFT_INPUT[i]/100); }
                        else { FFT_INPUT[i] = 0.0; }
                    }
}
```

```
////////////////////////////////////////////////////////////////////////////
// Host --> Target Command Analysis & Processing
int Command_Analysis()
{
		int i;
		unsigned int gain_ch, gain_data;
		unsigned int command;

		command = cget4_mail();// command read
		switch( command ) {
			case 0x10 :
				Trig_Mode = 0;
				Auto_Mode = 1;
				Stationary_Mode = 0;
				Average_Count = 1;			// Average Count
				circular_init();
				cad_start();		//  ad initialize & start
				break;
			case 0x11 :
				Trig_Mode = 1;
				Auto_Mode = 0;
				Stationary_Mode = 0;
				Average_Count = 1;			// Average Count
				circular_init();
				cad_start();		//  ad initialize & start
				break;
			case 0x12 :
				Trig_Mode = 0;
				Auto_Mode = 0;
				Stationary_Mode = 1;
				cad_start();		//  ad initialize & start
				break;
			case 0x13:
				cad_end();		//  ad end
				break;
			case 0x20 :
				break;
			case 0x30 :	// check point set
				command  = cget4_mail();		// Get channel no
				chk_data = command;
				break;
			case 0x31 :	// check point channel
				command  = cget4_mail();
				chk_channel = command;
				break;
			case 0x32 :	// check point channel
				command  = cget4_mail();
				pre_trigger = command;
				break;
```

```
case 0x33 :     //  reference channel )
                command  = cget4_mail();
                Trigger_Quantity_Reference = command;
                break;
case 0x34 :     // ( response channel )
                command  = cget4_mail();
                Trigger_Quantity_Response = command;
                break;
case 0x40 :     // gain control  1byte  0000   00  00
                                //      ch  :    data
                                //      ch = 1,2,4,8
                command = cget4_mail();                 // Get gain data
                gain_ch   = ( command & 0x000000f0 ) >> 4 ;
                gain_data =   command & 0x00000003;
                switch( gain_ch ) {
                                case 1 :
                                                gain_ctrl_ch0(gain_data);
                                                break;
                                case 2 :
                                                gain_ctrl_ch1(gain_data);
                                                break;
                                case 4 :
                                                gain_ctrl_ch2(gain_data);
                                                break;
                                case 8 :
                                                gain_ctrl_ch3(gain_data);
                                                break;
                                default :
                                                break;
                }
                break;
case 0x41 :
                break;
case 0x50 :
                c3xtimer();                             // timer setup
                break;
case 0x51 :
                c3xtimer_reset();    // timer setup
                break;
case 0x52 :
                command  = cget4_mail();                // Get channel no
                c3xtimer_init( command );               // timer setup(Period)
                break;
case 0x60 :
                command  = cget4_mail();                // Get channel no
                Winfor_refernce = command;              // timer setup(Period)
                break;
```

```
                    case 0x61 :
                                command  = cget4_mail();            // Get channel no
                                Winfor_response = command;          // timer setup(Period)
                                break;
                    case 0x62 :
                                command  = cget4_mail();            // Get channel no
                                Exp_Decay_Reference = command;// timer setup(Period)
                                break;
                    case 0x63 :
                                command  = cget4_mail();            // Get channel no
                                Exp_Decay_Response = command;// timer setup(Period)
                                break;
                    case 0x70 :
                                command  = cget4_mail();            // Get channel no
                                Frame_Size = command;               // timer setup(Period)
                                circular_init();
                                sin_table();
                                break;
                    case 0x71 :
                                command  = cget4_mail();            // Get channel no
                                Average_Count = command;            // timer setup(Period)
                                circular_init();
                                break;
                    case 0x72 :
                                command  = cget4_mail();            // Get channel no
                                Overlap = command;                  // timer setup(Period)
                                break;
                    // 0 : time. 1: time & window. 2: spectra, 3: time & spectra, 4: avg spectra
                    case 0x80 :
                                command  = cget4_mail();
                                Data_Processing = command;
                                break;
                    case 0xF0 :
                                cput_mail( 0xee );   // end
                                return 1;
                                break;
                    default :
                                break;
          }

          return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////
// Circular & Mode Variable Init.( Flag )

void ptr_init()
{
                    block_cnt   = 0 ;
                    block_start = 0 ;
                    exit_cnt    = 0 ;
                    chk_point   = 0 ;

                    Trig_Mode   = 0;
                    Auto_Mode   = 0;
                    Stationary_Mode = 0;

                    garbage_clear=0;
}

////////////////////////////////////////////////////////////////////////////
// Sin Table
int sin_table()
{
                    unsigned int i;
                    float theta;

                    theta = 2*PI/Frame_Size;
                    for (i=0;i<Half_FFT_Size;i++)   /* fill sin table in memory */
                                        SinTable[i]=sin(i*theta);

                    return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////
main()
{
                unsigned int i,j;
                unsigned int ad_flag;
                unsigned int mail_flag;
                unsigned int command;

                circular_init();  // configuration setup
                sin_table();
                ptr_init();
                c3xtimer();                             // timer setup

                while(1) {
                                ad_flag = cget_ad_flag(); // ad flag check,
                                if ( ad_flag != 0 ) {
                                                if ( garbage_clear == 0 ) {// garbage
                                                                cget_ad();
                                                                cget_ad();
                                                                garbage_clear = 1;
                                                                // memory point reset
                                                                cad_memory_reset();
                                                                continue;
                                                }
                                                cget_ad(); // FiFo data -> Memory
```

```
    if( Trig_Mode ) {
        if ( exit_cnt==0 ) {// check point
                        switch( chk_channel ) {
                            case 0 :
                                                for (i = 0; i < One_Read_Size; i++) {
                                                    chk_point = cget_ad_ch0(); // chekc point
                                                    chk_point = ( chk_point & 0x0000ffff) ;
                                                    if( chk_point >= 0x8000 ) chk_point = chk_point ^ 0x0000ffff;
                                                    if(( chk_data <=  chk_point) && ( chk_point <= 0x00007fff )) {
                                                                        exit_cnt    = pre_trigger; break;
                                                    }
                                                }
                                                break;
                            case 1 :
                                                for (i = 0; i < One_Read_Size; i++) {
                                                    chk_point = cget_ad_ch1(); // chekc point
                                                    chk_point = ( chk_point & 0x0000ffff) ;
                                                    if( chk_point >= 0x8000 ) chk_point = chk_point ^ 0x0000ffff;
                                                    if(( chk_data <=  chk_point ) && ( chk_point <= 0x00007fff )) {
                                                                        exit_cnt    = pre_trigger;break;
                                                    }
                                                }
                                                break;
                            case 2 :
                                                for (i = 0; i < One_Read_Size; i++) {
                                                    chk_point = cget_ad_ch2(); // chekc point
                                                    chk_point = ( chk_point & 0x0000ffff) ;
                                                    if( chk_point >= 0x8000 ) chk_point = chk_point ^ 0x0000ffff;
                                                    if(( chk_data <=  chk_point) && ( chk_point <= 0x00007fff )) {
                                                                        exit_cnt    = pre_trigger; break;
                                                    }
                                                }
                                                break;
                            case 3 :
                                                for (i = 0; i < One_Read_Size; i++) {
                                                    chk_point = cget_ad_ch3(); // chekc point
                                                    chk_point = ( chk_point & 0x0000ffff) ;
                                                    if( chk_point >= 0x8000 ) chk_point = chk_point ^ 0x0000ffff;
                                                    if(( chk_data <=  chk_point) && ( chk_point <= 0x00007fff )) {
                                                                        exit_cnt    = pre_trigger; break;
                                                    }
                                                }
                                                break;
                            default :
                                                break;
                        }
        }
        else { exit_cnt= exit_cnt + 1;      }
    }
    if( Auto_Mode || Stationary_Mode) { exit_cnt= exit_cnt + 1;    }
    if( Auto_Mode || Trig_Mode ||  Stationary_Mode) {
                        block_cnt = block_cnt + 1;
                        if ( block_cnt >= Circular_Size ) {         cad_memory_reset(); block_cnt = 0; }
    }
}
```

```
if ( exit_cnt >= Exit_Size) {
                cad_end();
                cad_memory_reset();                                     // memory point reset

                if( Trig_Mode ) {
                                // Circular Buffer Point

                                block_start = block_cnt;


                                // Data arrange
                                Data_Arrange( RAWBufferIn_CH0, FFTBufferIn_CH0 );
                                Data_Arrange( RAWBufferIn_CH1, FFTBufferIn_CH1);
                                Data_Arrange( RAWBufferIn_CH2, FFTBufferIn_CH2);
                                Data_Arrange( RAWBufferIn_CH3, FFTBufferIn_CH3);
                }
                if( Auto_Mode || Trig_Mode ) {
                                if( Auto_Mode ) {
                                                // Circular Buffer Point
                                                //
                                                //
                                                block_start = 0;

                                                // Data arrange
                                                Data_Arrange( RAWBufferIn_CH0, FFTBufferIn_CH0 );
                                                Data_Arrange( RAWBufferIn_CH1, FFTBufferIn_CH1);
                                                Data_Arrange( RAWBufferIn_CH2, FFTBufferIn_CH2);
                                                Data_Arrange( RAWBufferIn_CH3, FFTBufferIn_CH3);
                                }


                                // integer --> floating convertion
                                ////////////////////////////////////////////////////////////////
                                // Channel 0 ~ 3 convert
                                ////////////////////////////////////////////////////////////////
                                //RawToFFT( block_start, FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                switch( chk_channel ) {
                                                case 0 :
                                                   RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 1);
                                                   RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 0);
                                                   RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 0);
                                                   RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 0);
                                                   break;
                                                case 1 :
                                                   RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                                   RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 1);
                                                   RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 0);
                                                   RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 0);
                                                   break;
```

```
                                        case 2 :
                                            RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                            RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 0);
                                            RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 1);
                                            RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 0);
                                            break;
                                        case 3 :
                                            RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                            RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 0);
                                            RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 0);
                                            RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 1);
                                            break;
                                        default :
                                            break;
                    }
                    // Conversion End
                    ////////////////////////////////////////////////////////////
                    block_start = 0;  //
                    cput_mail( block_start );
}


if( Stationary_Mode ) {
                    for (i = 0; i < Average_Count; i++) {

                                        // Data arrange
                                        Data_Arrange_Stationary( RAWBufferIn_CH0, FFTBufferIn_CH0, i);
                                        Data_Arrange_Stationary( RAWBufferIn_CH1, FFTBufferIn_CH1, i);
                                        Data_Arrange_Stationary( RAWBufferIn_CH2, FFTBufferIn_CH2, i);
                                        Data_Arrange_Stationary( RAWBufferIn_CH3, FFTBufferIn_CH3, i);

                                        // integer --> floating convertion
                                        ////////////////////////////////////////////////////////////////
                                        // Channel 0 ~ 3 convert
                                        ////////////////////////////////////////////////////////////////
                                        //RawToFFT( block_start, FFTBufferIn_CH0, FFTToIEEE_CH0, 0)
                                        //1 = window reference(trigger channel), 0 = window response
                                        switch( chk_channel ) {
                                            case 0 :
                                                        RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 1);
                                                        RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 0);
                                                        RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 0);
                                                        RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 0);
                                                        break;
                                            case 1 :
                                                        RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                                        RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 1);
                                                        RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 0);
                                                        RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 0);
                                                        break;
```

```
                                    case 2 :
                                                RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                                RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 0);
                                                RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 1);
                                                RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 0);
                                                break;
                                    case 3 :
                                                RawToFFT( FFTBufferIn_CH0, FFTToIEEE_CH0, 0);
                                                RawToFFT( FFTBufferIn_CH1, FFTToIEEE_CH1, 0);
                                                RawToFFT( FFTBufferIn_CH2, FFTToIEEE_CH2, 0);
                                                RawToFFT( FFTBufferIn_CH3, FFTToIEEE_CH3, 1);
                                                break;
                                    default :
                                                break;
                                }
                                block_start = 0;  //
                                cput_mail( block_start );  //
                                while(1) {
                                    mail_flag = cget4_mail_flag();          // mail flag check
                                    if ( mail_flag == 0 ) continue; // skip

                                    command  = cget4_mail();
                                    if( command == 0x12 ) break;
                                }
                            }
                        }

                        ptr_init();
                    }
                    // host -> target command check
                    mail_flag = cget4_mail_flag();              // mail flag check
                    if ( mail_flag == 0 ) continue; // skip
                    if( Command_Analysis() ) exit(1);  // quit = 1
    }
}
```

B. Firmware Source code (FFT)

```
*******************************************************************************
* FILENAME          : ffft_rl.asm
*
* WRITTEN BY        : Alex Tessarolo
*                     Texas Instruments, Australia
*
* DATE              : 23rd July 1991
*
* VERSION           : 2.0
*
*******************************************************************************
* VER                 DATE                            COMMENTS
* ---                 ----                            --------
* 1.0                 18th July 91                    Original Release.
* 2.0                 23rd July 91                    Most Stages Modified.
*                                                     Minimum FFT Size increased from 32 to 64.
*                                                     Faster in place bit reversing algorithm.
*                                                     Program size increased by about 100 words.
*                                                     One extra data word required.
*
*******************************************************************************
*
* SYNOPSIS:         int                 ffft_rl( FFT_SIZE, LOG_SIZE, SOURCE_ADDR, DEST_ADDR,
*                                                    SINE_TABLE, BIT_REVERSE );
*
*                                       int         FFT_SIZE        ; 64, 128, 256, 512, 1024, ...
*                                       int         LOG_SIZE        ; 6,  7,  8,  9,  10, ...
*                                       float       *SOURCE_ADDR    ; Points to location of source data.
*                                       float       *DEST_ADDR      ; Points to where data will be
*                                                                   ; operated on and stored.
*                                       float       *SINE_TABLE     ; Points to the SIN/COS table.
*                                       int         BIT_REVERSE     ; =  0, Bit Reversing is disabled.
*                                                                   ; <> 0, Bit Reversing is enabled.
*
*                                       NOTE:       1) If SOURCE_ADDR = DEST_ADDR, then in place bit
*                                                     reversing is performed, if enabled (more
*                                                     processor intensive).
*                                                   2) FFT_SIZE must be >= 64 (this is not checked).
*
* DESCRIPTION:      Generic function to do a radix-2 FFT computation on the C30.
*                                       The data array is FFT_SIZE-long with only real data. The
*                                       output is stored in the same locations with real and imaginary
*                                       points R and I as follows:
*
*                                                   DEST_ADDR[0]                -> R(0)
*                                                                                  R(1)
*                                                                                  R(2)
*                                                                                  R(3)
*                                                                                  .
*                                                                                  .
*                                                                                  R(FFT_SIZE/2)
*                                                                                  I(FFT_SIZE/2 - 1)
*                                                                                  .
*                                                                                  .
*                                                                                  I(2)
*                                                   DEST_ADDR[FFT_SIZE - 1]     -> I(1)
*
*                                       The program is based on the FORTRAN program in the paper
*                                       by Sorensen et al., June 1987 issue of Trans. on ASSP.
*
*                                       Bit reversal is optionally implemented at the beginning
*                                       of the function.
*
```

```
*                                       The sine/cosine table for the twiddle factors is expected
*                                       to be supplied in the following format:
*
*                                       SINE_TABLE[0]                   -> sin(0*2*pi/FFT_SIZE)
*                                                                          sin(1*2*pi/FFT_SIZE)
*                                                                   .
*                                                                   .
*                                                                   .
*                                                                          sin((FFT_SIZE/2 -2)*2*pi/FFT_SIZE)
*                                       SINE_TABLE[FFT_SIZE/2 - 1] -> sin((FFT_SIZE/2 -1)*2*pi/FFT_SIZE)
*
*                                       NOTE: The table is the first half period of a sine wave.
*
*                                       Stack structure upon call:
*
*                                       +------------------+
*                                       -FP(7)  | BIT_REVERSE    |
*                                       -FP(6)  | SINE_TABLE     |
*                                       -FP(5)  | DEST_ADDR      |
*                                       -FP(4)  | SOURCE_ADDR    |
*                                       -FP(3)  | LOG_SIZE       |
*                                       -FP(2)  | FFT_SIZE       |
*                                       -FP(1)  | return addr    |
*                                       -FP(0)  | old FP         |
*                                        +------------------+
*
*************************************************************************
*
*                                       NOTE:           Calling C program can be compiled using either large
*                                                       or small model.
*
*                                       WARNING: DP initialised only once in the program. Be wary
*                                                       with interrupt service routines. Make sure interrupt
*                                                       service routines save the DP pointer.
*
*                                       WARNING: The DEST_ADDR must be aligned such that the first
*                                                       LOG_SIZE bits are zero (this is not checked by the
*                                                       program).
*
*************************************************************************
*
* REGISTERS USED: R0, R1, R2, R3, R4, R5, R6, R7
*                                         AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7
*                                         IR0, IR1
*                                         RC, RS, RE
*                                         DP
*
* MEMORY REQUIREMENTS:            Program = 405 Words (approximately)
*                                 Data    =   7 Words
*                                 Stack   =  12 Words
*
*************************************************************************
*
* BENCHMARKS:   Assumptions       - Program in RAM0
*                                 - Reserved data in RAM0
*                                 - Stack on Primary/Expansion Bus RAM
*                                 - Sine/Cosine tables in RAM0
*                                 - Processing and data destination in RAM1.
*                                 - Primary/Expansion Bus RAM, 0 wait state.
*
*               FFT Size          Bit Reversing     Data Source       Cycles(C30)
*               --------          -------------     -----------       -----------
*               1024              OFF               RAM1              19816 approx.
*Note           : This number does not include the C callable overheads.
*                 Add 57 cycles for these overheads.
*
```

```
*******************************************************************************

FP              .set            AR3

                .global         _ffft_rl        ; Entry execution point.

FFT_SIZE:       .usect          ".fftdata",1    ; Reserve memory for arguments.
LOG_SIZE:       .usect          ".fftdata",1
SOURCE_ADDR:    .usect          ".fftdata",1
DEST_ADDR:      .usect          ".fftdata",1
SINE_TABLE:     .usect          ".fftdata",1
BIT_REVERSE:    .usect          ".fftdata",1
SEPARATION:     .usect          ".fftdata",1

;
; Initialise C Function.
;
                .sect           ".ffttext"
_ffft_rl:       PUSH            FP                      ; Preserve C environment.
                LDI             SP,FP
                PUSH            R4
                PUSH            R5
                PUSH            R6
                PUSHF           R6
                PUSH            R7
                PUSHF           R7
                PUSH            AR4
                PUSH            AR5
                PUSH            AR6
                PUSH            AR7
                PUSH            DP

                LDP             FFT_SIZE        ; Initialise DP pointer.

                LDI             *-FP(2),R0      ; Move arguments from stack.
                STI             R0,@FFT_SIZE
                LDI             *-FP(3),R0
                STI             R0,@LOG_SIZE
                LDI             *-FP(4),R0
                STI             R0,@SOURCE_ADDR
                LDI             *-FP(5),R0
                STI             R0,@DEST_ADDR
                LDI             *-FP(6),R0
                STI             R0,@SINE_TABLE
                LDI             *-FP(7),R0
                STI             R0,@BIT_REVERSE

;
; Check Bit Reversing Mode (on or off).
;
; BIT_REVERSING =  0, then OFF (no bit reversing).
; BIT_REVERSING <> 0, Then ON.
;
                LDI             @BIT_REVERSE,R0
                CMPI            0,R0
                BZ              MOVE_DATA
```

```
;
; Check Bit Reversing Type.
;
; If SourceAddr =  DestAddr, Then In Place Bit Reversing.
; If SourceAddr <> DestAddr, Then Standard Bit Reversing.
;
                LDI     @SOURCE_ADDR,R0
                CMPI    @DEST_ADDR,R0
                BEQ     IN_PLACE

;
; Bit reversing Type 1 (From Source to Destination).
;
; NOTE: abs(SOURCE_ADDR - DEST_ADDR) must be > FFT_SIZE, this is not checked.
;
                LDI     @FFT_SIZE,R0
                SUBI    2,R0
                LDI     @FFT_SIZE,IR0
                LSH     -1,IR0                  ; IRO = Half FFT size.
                LDI     @SOURCE_ADDR,AR0
                LDI     @DEST_ADDR,AR1

                LDF     *AR0++,R1

                RPTS    R0
                LDF     *AR0++,R1
          ||    STF     R1,*AR1++(IR0)B

                STF     R1,*AR1++(IR0)B

                BR      START

;
; In Place Bit Reversing.
;
                ; Bit Reversing On Even Locations, 1st Half Only.

IN_PLACE:       LDI     @FFT_SIZE,IR0
                LSH     -2,IR0                  ; IRO = Quarter FFT size.
                LDI     2,IR1

                LDI     @FFT_SIZE,RC
                LSH     -2,RC
                SUBI    3,RC
                LDI     @DEST_ADDR,AR0
                LDI     AR0,AR1
                LDI     AR0,AR2

                NOP     *AR1++(IR0)B
                NOP     *AR2++(IR0)B
                LDF     *++AR0(IR1),R0
                LDF     *AR1,R1
                CMPI    AR1,AR0                 ; Xchange Locations only if AR0<AR1.
                LDFGT   R0,R1
                LDFGT   *AR1++(IR0)B,R1

                RPTB    BITRV1
                LDF     *++AR0(IR1),R0
          ||    STF     R0,*AR0
                LDF     *AR1,R1
          || STF        R1,*AR2++(IR0)B
                CMPI    AR1,AR0
                LDFGT   R0,R1
```

```
BITRV1:         LDFGT       *AR1++(IR0)B,R0

                STF         R0,*AR0
                STF         R1,*AR2

                ; Perform Bit Reversing On Odd Locations, 2nd Half Only.

                LDI         @FFT_SIZE,RC
                LSH         -1,RC
                LDI         @DEST_ADDR,AR0
                ADDI        RC,AR0
                ADDI        1,AR0
                LDI         AR0,AR1
                LDI         AR0,AR2
                LSH         -1,RC
                SUBI        3,RC

                NOP         *AR1++(IR0)B
                NOP         *AR2++(IR0)B
                LDF         *++AR0(IR1),R0
                LDF         *AR1,R1
                CMPI        AR1,AR0                 ; Xchange Locations only if AR0<AR1.
                LDFGT       R0,R1
                LDFGT       *AR1++(IR0)B,R1

                RPTB        BITRV2
                LDF         *++AR0(IR1),R0
          || STF            R0,*AR0
                LDF         *AR1,R1
          || STF            R1,*AR2++(IR0)B
                CMPI        AR1,AR0
                LDFGT       R0,R1
BITRV2:         LDFGT       *AR1++(IR0)B,R0

                STF         R0,*AR0
                STF         R1,*AR2

                ; Perform Bit Reversing On Odd Locations, 1st Half Only.

                LDI         @FFT_SIZE,RC
                LSH         -1,RC
                LDI         RC,IR0
                LDI         @DEST_ADDR,AR0
                LDI         AR0,AR1
                ADDI        1,AR0
                ADDI        IR0,AR1
                LSH         -1,RC
                LDI         RC,IR0
                SUBI        2,RC

                LDF         *AR0,R0
                LDF         *AR1,R1

                RPTB        BITRV3
                LDF         *++AR0(IR1),R0
          || STF  R0,*AR1++(IR0)B
BITRV3:         LDF         *AR1,R1
          || STF  R1,*-AR0(IR1)

                STF         R0,*AR1
                STF         R1,*AR0

                BR          START
```

```
;
; Check Data Source Locations.
;
; If SourceAddr =  DestAddr, Then do nothing.
; If SourceAddr <> DestAddr, Then move data.
;

MOVE_DATA:      LDI             @SOURCE_ADDR,R0
                CMPI            @DEST_ADDR,R0
                BEQ             START

                LDI             @FFT_SIZE,R0
                SUBI            2,R0
                LDI             @SOURCE_ADDR,AR0
                LDI             @DEST_ADDR,AR1

                LDF             *AR0++,R1

                RPTS            R0
                LDF             *AR0++,R1
            ||  STF             R1,*AR1++

                STF             R1,*AR1
```

```
;
; Perform first and second FFT loops.
;   _                 _____
;   |  AR1 →          |__I1__| 0 ←   [X(I1) + X(I2)] + [X(I3) + X(I4)]
;   |  AR2 →          |__I2__| 1 ←   [X(I1) - X(I2)]
;   |  AR3 →          |__I3__| 2 ←   [X(I1) + X(I2)] - [X(I3) + X(I4)]
;   |_ AR4 →          |__I4__| 3 ← -[X(I3) - X(I4)]
;      AR1 →          |_____|4
;                     |  .  |
;                  .
```

⇩

```
START:      LDI     @DEST_ADDR,AR1
            LDI     AR1,AR2
            LDI     AR1,AR3
            LDI     AR1,AR4
            ADDI    1,AR2
            ADDI    2,AR3
            ADDI    3,AR4
            LDI     4,IR0
            LDI     @FFT_SIZE,RC
            LSH     -2,RC
            SUBI    2,RC

            LDF     *AR2,R0             ; R0 = X(I2)
        || LDF      *AR3,R1             ; R1 = X(I3)
            ADDF3   R1,*AR4,R4          ; R4 = X(I3) + X(I4)
            SUBF3   R1,*AR4++(IR0),R5; R5 = -[X(I3) - X(I4)] --+
            SUBF3   R0,*AR1,R6          ; R6 = X(I1) - X(I2) --+    |
            ADDF3   R0,*AR1++(IR0),R7; R7 = X(I1) + X(I2)    |   |
            ADDF3   R7,R4,R2            ; R2 = R7 + R4 ----+ |   |
            SUBF3   R4,R7,R3            ; R3 = R7 - R4 --+ ||   |
                                        ;                | | | |
            RPTB    LOOP1_2             ;                  | | | |
            LDF     *+AR2(IR0),R0       ;                    | | | |
        || LDF      *+AR3(IR0),R1       ;                    | | | |
            ADDF3   R1,*AR4,R4          ;                    | | | |
        || STF  R3,*AR3++(IR0)          ; X(I3) <-------+  | | |
            SUBF3   R1,*AR4++(IR0),R5;                     | | |
        || STF  R5,*-AR4(IR0)           ; X(I4) <------  --|-|-+
            SUBF3   R0,*AR1,R6          ;                    | |
        || STF      R6,*AR2++(IR0)      ; X(I2) <---------|-+
            ADDF3   R0,*AR1++(IR0),R7;                     |
        || STF      R2,*-AR1(IR0)       ; X(I1) <-------- +
            ADDF3   R7,R4,R2
LOOP1_2:    SUBF3   R4,R7,R3

            STF     R3,*AR3
        || STF  R5,*-AR4(IR0)
            STF     R6,*AR2
        || STF      R2,*-AR1(IR0)
```

```
;
; Perform Third FFT Loop.
;
; Part A:
;   +-                  _______
;   | AR1 →             |__I1__| 0 ←  X(I1) + X(I3)
;   |                   |______| 1
;   |                   |__I2__| 2
;   |                   |______| 3
;   | AR2 →             |__I3__| 4 ←  X(I1) - X(I3)
;   |                   |______| 5
;   | AR3 →             |__I4__| 6 ← -X(I4)
;   +-                  |______| 7
;     AR1 →             |______| 8
;                       |______| 9
;                       |  .   |
;                     .
;                     .
;
                      LDI       @DEST_ADDR,AR1
                      LDI       AR1,AR2
                      LDI       AR1,AR3
                      ADDI      4,AR2
                      ADDI      6,AR3
                      LDI       8,IR0
                      LDI       @FFT_SIZE,RC
                      LSH       -3,RC
                      SUBI      2,RC

                      SUBF3     *AR2,*AR1,R1
                      ADDF3     *AR2,*AR1,R2
                      NEGF      *AR3,R3

                      RPTB      LOOP3_A
                      LDF       *+AR2(IR0),R0     ; R0 =  X(I3)
               || STF           R2,*AR1++(IR0)
                      SUBF3     R0,*AR1,R1        ; R1 =  X(I1) - X(I3) -----+
               || STF           R1,*AR2++(IR0)    ;                         |
                      ADDF3     R0,*AR1,R2        ; R2 =  X(I1) + X(I3) --+  |
               || STF           R3,*AR3++(IR0)    ;                       |  |
LOOP3_A:              NEGF      *AR3,R3           ; R3 = -X(I4) --+       |  |
                                                  ;               |       |  |
                      STF       R2,*AR1           ; X(I1) <-------+-------+  |
                || STF           R1,*AR2          ; X(I3) <-------+----------+
                      STF       R3,*AR3           ; X(I4) <-------+
```

```
;
; Part B:
;    +--                 _______
;    |                  |_______| 0
;    |AR0 →             |__I1___| 1 ←  X(I1) + [X(I3)*COS + X(I4)*COS]
;    |                  |_______| 2
;    |AR1 →             |__I2___| 3 ←  X(I1) - [X(I3)*COS + X(I4)*COS]
;    |                  |_______| 4
;    |AR2 →             |__I3___| 5 ← X(I2) - [X(I3)*COS - X(I4)*COS]
;    |                  |_______| 6
;    |AR3 →             |__I4___| 7 ←  X(I2) - [X(I3)*COS - X(I4)*COS]
;    +-                 |_______| 8
;      AR0 →            |_______| 9              NOTE: COS(2*pi/8) = SIN(2*pi/8)
;                       |   .   |
;                    .
;                    .
;
;
                        LDI             @FFT_SIZE,RC
                        LSH             -3,RC
                        LDI             RC,IR1
                        SUBI            3,RC
                        LDI             8,IR0
                        LDI             @DEST_ADDR,AR0
                        LDI             AR0,AR1
                        LDI             AR0,AR2
                        LDI             AR0,AR3
                        ADDI            1,AR0
                        ADDI            3,AR1
                        ADDI            5,AR2
                        ADDI            7,AR3

                        LDI             @SINE_TABLE,AR7                 ; Initialise table pointers.

                        LDF             *++AR7(IR1),R7      ; R7   = COS(2*pi/8)
                                                            ; *AR7 = COS(2*pi/8)
                        MPYF3           *AR7,*AR2,R0        ; R0 =  X(I3)*COS

                        MPYF3           *AR3,R7,R1          ; R5 =  X(I4)*COS
                        ADDF3           R0,R1,R2            ; R2 =  [X(I3)*COS + X(I4)*COS]
                        MPYF3           *AR7,*+AR2(IR0),R0
                || SUBF3                R0,R1,R3            ; R3 = -[X(I3)*COS -  X(I4)*COS]
                        SUBF3           *AR1,R3,R4          ; R4 = -X(I2) + R3 --+
                        ADDF3           *AR1,R3,R4          ; R4 =  X(I2) + R3 ---|--+
                || STF  R4,*AR2++(IR0)  ; X(I3) <----------+    |
                        SUBF3           R2,*AR0,R4          ; R4 =  X(I1) - R2 ---+    |
                || STF  R4,*AR3++(IR0)  ; X(I4) <---------- |--+
                        ADDF3           *AR0,R2,R4          ; R4 =  X(I1) + R2 ---|--+
                || STF  R4,*AR1++(IR0)  ; X(I2) <----------+   |
                                                            ;                          |
                        RPTB            LOOP3_B             ;                          |
                        MPYF3           *AR3,R7,R1          ;                          |
                || STF  R4,*AR0++(IR0)  ; X(I1) <-------------+
                         ADDF3          R0,R1,R2
                        MPYF3           *AR7,*+AR2(IR0),R0
                || SUBF3                R0,R1,R3
                        SUBF3           *AR1,R3,R4
                        ADDF3           *AR1,R3,R4
                || STF  R4,*AR2++(IR0)
                        SUBF3           R2,*AR0,R4
                || STF  R4,*AR3++(IR0)
```

```
LOOP3_B:        ADDF3       *AR0,R2,R4
        || STF  R4,*AR1++(IR0)

                MPYF3       *AR3,R7,R1
        || STF  R4,*AR0++(IR0)
                ADDF3       R0,R1,R2
                SUBF3       R0,R1,R3
                SUBF3       *AR1,R3,R4
                ADDF3       *AR1,R3,R4
        || STF  R4,*AR2
                SUBF3       R2,*AR0,R4
        || STF  R4,*AR3
                ADDF3       *AR0,R2,R4
        || STF  R4,*AR1

                STF         R4,*AR0
```

```
;
; Perform Fourth FFT Loop.
;
; Part A:
;    +--                 _______
;    | AR1 →             |__I1__| 0 ←  X(I1) + X(I3)
;    |                   |______| 1
;    |                   |______| 2
;    |                   |______| 3
;    |                   |__I2__| 4
;    |                   |______| 5
;    |                   |______| 6
;    |                   |______| 7
;    | AR2 →             |__I3__| 8 ←  X(I1) - X(I3)
;    |                   |______| 9
;    |                   |______| 10
;    |                   |______| 11
;    | AR3 →             |__I4__| 12 ← X(I4)
;    |                   |______| 13
;    |                   |______| 14
;    +--                 |______| 15
;      AR1 →             |__I5__| 16
;                        |______| 17
;                        |  .   |
;                  .
;                  .
;

                        LDI     @DEST_ADDR,AR1
                        LDI     AR1,AR2
                        LDI     AR1,AR3
                        ADDI    8,AR2
                        ADDI    12,AR3
                        LDI     16,IR0
                        LDI     @FFT_SIZE,RC
                        LSH     -4,RC
                        SUBI    2,RC

                        SUBF3   *AR2,*AR1,R1
                        ADDF3   *AR2,*AR1,R2
                        NEGF    *AR3,R3

                        RPTB    LOOP4_A
                        LDF     *+AR2(IR0),R0       ; R0 =  X(I3)
                || STF  R2,*AR1++(IR0)
                        SUBF3   R0,*AR1,R1          ; R1 =  X(I1) - X(I3) -----+
                || STF  R1,*AR2++(IR0)              ;                          |
                        ADDF3   R0,*AR1,R2          ; R2 =  X(I1) + X(I3) --+  |
                || STF  R3,*AR3++(IR0)              ;                       |  |
LOOP4_A:                NEGF    *AR3,R3             ; R3 = -X(I4) ---+      |  |
                                                    ;                |      |  |
                        STF     R2,*AR1             ; X(I1) <--------|---+  |
                || STF  R1,*AR2             ; X(I3) <--------|------+
                        STF     R3,*AR3             ; X(I4) <--------+
```

```
;
; Part B:
;    +-                 _____________
;    |                  |_____________| 0
;    |AR0  →            |__I1_(3rd)__| 1  ←  X(I1) + [X(I3)*COS + X(I4)*SIN]
;    |                  |__I1_(2nd)_| 2      .
;    |                  |__I1_(1st)__| 3      .
;    |                  |_____________| 4
;    |                  |__I2_(1st)__| 5      .
;    |                  |__I2_(2nd)_| 6      .
;    | AR1→             |__I2_(3rd)__| 7  ←  X(I1) - [X(I3)*COS + X(I4)*SIN]
;    |                  |_____________| 8
;    | AR2→             |__I3_(3rd)_)| 9  ← X(I2) - [X(I3)*SIN - X(I4)*COS]
;    |                  |__I3_(2nd)_| 10     .
;    | AR4→             |__I3_(1st)__| 11     .
;    |                  |_____________| 12
;    |                  |__I4_(1st)__| 13     .
;    |                  |__I4_(2nd)_| 14     .
;    | AR3 →            |__I4_(3rd)_| 15 ←  X(I2) - [X(I3)*SIN -  X(I4)*COS]
;    +=                 |_____________| 16
;      AR0→             |_____________| 17
;                       |    .        |
;                      .
;                      .
;
;
                        LDI             @FFT_SIZE,RC
                        LSH             -4,RC
                        LDI             RC,IR1
                        LDI             2,IR0
                        SUBI            3,RC
                        LDI             @DEST_ADDR,AR0
                        LDI             AR0,AR1
                        LDI             AR0,AR2
                        LDI             AR0,AR3
                        LDI             AR0,AR4
                        ADDI            1,AR0
                        ADDI            7,AR1
                        ADDI            9,AR2
                        ADDI            15,AR3
                        ADDI    11,AR4

                        LDI             @SINE_TABLE,AR7
                        LDF             *++AR7(IR1),R7      ; R7 = SIN(1*[2*pi/16])
                                                            ; *AR7 = COS(3*[2*pi/16])
                        LDI             AR7,AR6
                        LDF             *++AR6(IR1),R6      ; R6 = SIN(2*[2*pi/16])
                                                            ; *AR6 = COS(2*[2*pi/16])
                        LDI             AR6,AR5
                        LDF             *++AR5(IR1),R5      ; R5 = SIN(3*[2*pi/16])
                                                            ; *AR5 = COS(1*[2*pi/16])
                        LDI             16,IR1
                        MPYF3           *AR7,*AR4,R0        ; R0 =  X(I3)*COS(3)
```

```
        MPYF3       *++AR2(IR0),R5,R4; R4 =  X(I3)*SIN(3)
        MPYF3       *--AR3(IR0),R5,R1; R1 =  X(I4)*SIN(3)
        MPYF3       *AR7,*AR3,R0        ; R0 =  X(I4)*COS(3)
 || ADDF3           R0,R1,R2            ; R2 =  [X(I3)*COS + X(I4)*SIN]
        MPYF3       *AR6,*-AR4,R0
|| SUBF3            R4,R0,R3            ; R3 = -[X(I3)*SIN - X(I4)*COS]
        SUBF3       *--AR1(IR0),R3,R4; R4 = -X(I2) + R3 --+
        ADDF3       *AR1,R3,R4          ; R4 =  X(I2) + R3 ---|--+
|| STF   R4,*AR2--    ; X(I3) <----------+    |
        SUBF3       R2,*++AR0(IR0),R4; R4 =  X(I1) - R2 ---+    |
|| STF   R4,*AR3     ; X(I4) <-----------|--+
        ADDF3       *AR0,R2,R4          ; R4 =  X(I1) + R2 --- |--+
|| STF   R4,*AR1     ; X(I2) <------------+  |
                                              ;                  |
        MPYF3       *++AR3,R6,R1      ;                        |
|| STF   R4,*AR0     ; X(I1) <---------------+
        ADDF3       R0,R1,R2
        MPYF3       *AR5,*-AR4(IR0),R0
|| SUBF3            R0,R1,R3
        SUBF3       *++AR1,R3,R4
        ADDF3       *AR1,R3,R4
|| STF   R4,*AR2
        SUBF3       R2,*--AR0,R4
|| STF   R4,*AR3
        ADDF3       *AR0,R2,R4
|| STF   R4,*AR1

        MPYF3       *--AR2,R7,R4
|| STF   R4,*AR0
        MPYF3       *++AR3,R7,R1
        MPYF3       *AR5,*AR3,R0
|| ADDF3            R0,R1,R2
        MPYF3       *AR7,*++AR4(IR1),R0
|| SUBF3            R4,R0,R3
        SUBF3       *++AR1,R3,R4
        ADDF3       *AR1,R3,R4
|| STF   R4,*AR2++(IR1)
        SUBF3       R2,*--AR0,R4
|| STF   R4,*AR3++(IR1)
        ADDF3       *AR0,R2,R4
|| STF   R4,*AR1++(IR1)

        RPTB        LOOP4_B
        MPYF3       *++AR2(IR0),R5,R4
|| STF   R4,*AR0++(IR1)
        MPYF3       *--AR3(IR0),R5,R1
        MPYF3       *AR7,*AR3,R0
|| ADDF3            R0,R1,R2
        MPYF3       *AR6,*-AR4,R0
|| SUBF3            R4,R0,R3
        SUBF3       *--AR1(IR0),R3,R4
        ADDF3       *AR1,R3,R4
|| STF   R4,*AR2 —

        SUBF3       R2,*++AR0(IR0),R4
|| STF   R4,*AR3
        ADDF3       *AR0,R2,R4
|| STF   R4,*AR1

        MPYF3       *++AR3,R6,R1
|| STF   R4,*AR0
        ADDF3       R0,R1,R2
        MPYF3       *AR5,*-AR4(IR0),R0
|| SUBF3            R0,R1,R3
```

```
            SUBF3       *++AR1,R3,R4
            ADDF3       *AR1,R3,R4
    || STF  R4,*AR2
            SUBF3       R2,*--AR0,R4
    || STF  R4,*AR3
            ADDF3       *AR0,R2,R4
    || STF  R4,*AR1
            MPYF3       *--AR2,R7,R4
    || STF  R4,*AR0
            MPYF3       *++AR3,R7,R1
            MPYF3       *AR5,*AR3,R0
    || ADDF3            R0,R1,R2
            MPYF3       *AR7,*++AR4(IR1),R0
    || SUBF3            R4,R0,R3
            SUBF3       *++AR1,R3,R4
            ADDF3       *AR1,R3,R4
    || STF  R4,*AR2++(IR1)
            SUBF3       R2,*--AR0,R4
    || STF  R4,*AR3++(IR1)
LOOP4_B:    ADDF3       *AR0,R2,R4
    || STF  R4,*AR1++(IR1)

            MPYF3       *++AR2(IR0),R5,R4
    || STF  R4,*AR0++(IR1)
            MPYF3       *--AR3(IR0),R5,R1
            MPYF3       *AR7,*AR3,R0
    || ADDF3            R0,R1,R2
            MPYF3       *AR6,*-AR4,R0
    || SUBF3            R4,R0,R3
            SUBF3       *--AR1(IR0),R3,R4
            ADDF3       *AR1,R3,R4
    || STF  R4,*AR2--
            SUBF3       R2,*++AR0(IR0),R4
    || STF  R4,*AR3
            ADDF3       *AR0,R2,R4
    || STF  R4,*AR1

            MPYF3       *++AR3,R6,R1
    || STF  R4,*AR0
            ADDF3       R0,R1,R2
            MPYF3       *AR5,*-AR4(IR0),R0
    || SUBF3            R0,R1,R3
            SUBF3       *++AR1,R3,R4
            ADDF3       *AR1,R3,R4
    || STF  R4,*AR2
            SUBF3       R2,*--AR0,R4
    || STF  R4,*AR3
            ADDF3       *AR0,R2,R4
    || STF  R4,*AR1

            MPYF3       *--AR2,R7,R4
    || STF  R4,*AR0
            MPYF3       *++AR3,R7,R1
            MPYF3       *AR5,*AR3,R0
    || ADDF3            R0,R1,R2
            SUBF3       R4,R0,R3
            SUBF3       *++AR1,R3,R4
            ADDF3       *AR1,R3,R4
    || STF  R4,*AR2
            SUBF3       R2,*--AR0,R4
    || STF  R4,*AR3
            ADDF3       *AR0,R2,R4
    || STF  R4,*AR1

            STF         R4,*AR0
```

```
;
; Perform Remaining FFT loops (loop 4 onwards).
;
;                                        LOOP
;                                        1st  2nd .....
;   +--          _____________
;   |           |____X'(I1)____|     0    0  ←  X'(I1) + X'(I3)
;   | AR1 →     |__X(I1)_(1st)_|     1    1  ←  X(I1) + [X(I3)*COS + X(I4)*SIN]
;   |           |__X(I1)_(2nd)_|     2    2     .
;   |           |__x(I1)_(3rd)_|     3    3     .
;   |           |      .       |
;   |           .
;   |   A →     |______________|
;   |           |____X'(I2)____|     8    16
;   |   B →     |      .       |
;   |           .
;   |           |______________|
;   |           |__X(I2)_(3rd)_|     13   29    .
;   |           |__X(I2)_(2nd)_|     14   30    .
;   | AR2 →     |__X(I2)_(1st)_|     15   31 ←  X(I1) - [X(I3)*COS + X(I4)*SIN]
;   |           |____X'(I3)____|     16   32 ←  X'(I1) - X'(I3)
;   | AR3 →     |__X(I3)_(1st)_|     17   33 ← -X(I2) - [X(I3)*SIN - X(I4)*COS]
;   |           |__X(I3)_(2nd)_|     18   34    .
;   |           |__X(I3)_(3rd)_|     19   35    .
;   |           |      .       |
;   |           .
;   |   C →     |______________|
;   |           |____X'(I4)____|     24   48 ← -X'(I4)
;   |   D →     |      .       |
;   |           .
;   |           |______________|
;   |           |__X(I4)_(3rd)_|     29   61    .
;   |           |__X(I4)_(2nd)_|     30   62    .
;   | AR4 →     |__X(I4)_(1st)__|    31   63 ←  X(I2) - [X(I3)*SIN - X(I4)*COS]
;   +--         |______________|     32   64
;     AR1 →     |______________|     33   65
;               |      .     |
;                .
;                .
;
;
                LDI     @FFT_SIZE,IR0
                LSH     -2,IR0
                STI     IR0,@SEPARATION
                LSH     -2,IR0
                LDI     5,R5
                LDI     3,R7
                LDI     16,R6
                LDI     @DEST_ADDR,AR5
                LDI     @DEST_ADDR,AR1
                LSH     -1,IR0
                LSH     1,R7
LOOP:           ADDI    1,R7
                LSH     1,R6
                LDI     AR1,AR4
                ADDI    R7,AR1          ; AR1 points at A.
                LDI     AR1,AR2
                ADDI    2,AR2           ; AR2 points at B.
                ADDI    R6,AR4
                SUBI    R7,AR4          ; AR4 points at D.
                LDI     AR4,AR3
                SUBI    2,AR3           ; AR3 points at C.
                LDI     @SINE_TABLE,AR0 ; AR0 points at SIN/COS table.
                LDI     R7,IR1
                LDI     R7,RC
```

```
INLOP:      ADDF3   *--AR1(IR1),*++AR2(IR1),R0      ; R0 = X'(I1) + X'(I3) ---+
            SUBF3   *--AR3(IR1),*AR1++,R1          ; R1 = X'(I1) - X'(I3) -+  |
            NEGF    *--AR4,R2                      ; R2 = -X'(I4) --+     | |
        || STF  R0,*-AR1                ; X'(I1) <------|---|-+
            STF     R1,*AR2--                      ; X'(I3) <------|---+
        || STF  R2,*AR4++(IR1)          ; X'(I4) <------+

            LDI     @SEPARATION,IR1; IR1=SEPARATION BETWEEN SIN/COS TBLS
            SUBI    3,RC

            MPYF3   *++AR0(IR0),*AR4,R4            ; R4 = X(I4)*SIN
            MPYF3   *AR0,*++AR3,R1                 ; R1 = X(I3)*SIN
            MPYF3   *++AR0(IR1),*AR4,R0            ; R0 = X(I4)*COS
            MPYF3   *AR0,*AR3,R0                   ; R0 = X(I3)*COS
        || SUBF3    R1,R0,R3                       ; R3 = -[X(I3)*SIN - X(I4)*COS]
            MPYF3   *++AR0(IR0),*-AR4,R0
        || ADDF3    R0,R4,R2                       ; R2 = X(I3)*COS + X(I4)*SIN
            SUBF3   *AR2,R3,R4                     ; R4 = R3 - X(I2) --*
            ADDF3   *AR2,R3,R4                     ; R4 = R3 + X(I2) --|--*
        || STF  R4,*AR3++               ; X(I3) <---------*    |
            SUBF3   R2,*AR1,R4                     ; R4 = X(I1) - R2 --*    |
        || STF  R4,*AR4--               ; X(I4) <---------|--*
            ADDF3   *AR1,R2,R4                     ; R4 = X(I1) + R2 --|--*
        || STF  R4,*AR2--               ; X(I2) <---------*    |
                                                   ;                        |
            RPTB    IN_BLK              ;                      |
            LDF     *-AR0(IR1),R3                  ;                        |
            MPYF3   *AR4,R3,R4          ;                      |
        || STF  R4,*AR1++               ; X(I1) <-------------*
            MPYF3   *AR3,R3,R1
            MPYF3   *AR0,*AR3,R0
        || SUBF3    R1,R0,R3
            MPYF3   *++AR0(IR0),*-AR4,R0
        || ADDF3    R0,R4,R2
            SUBF3   *AR2,R3,R4
            ADDF3   *AR2,R3,R4
        || STF  R4,*AR3++
            SUBF3   R2,*AR1,R4
        || STF  R4,*AR4--
```

```
IN_BLK:         ADDF3   *AR1,R2,R4
        || STF  R4,*AR2--

                LDF     *-AR0(IR1),R3
                MPYF3   *AR4,R3,R4
        || STF  R4,*AR1++
                MPYF3   *AR3,R3,R1
                MPYF3   *AR0,*AR3,R0
        || SUBF3        R1,R0,R3
                LDI     R6,IR1
    ADDF3       R0,R4,R2
                SUBF3   *AR2,R3,R4
                ADDF3   *AR2,R3,R4
        || STF  R4,*AR3++(IR1)
                SUBF3   R2,*AR1,R4
        || STF  R4,*AR4++(IR1)
                ADDF3   *AR1,R2,R4
        || STF  R4,*AR2++(IR1)

                STF     R4,*AR1++(IR1)

                SUBI3   AR5,AR1,R0
                CMPI    @FFT_SIZE,R0
                BLTD    INLOP                   ; LOOP BACK TO THE INNER LOOP
                LDI     @SINE_TABLE,AR0         ; AR0 POINTS TO SIN/COS TABLE
                LDI     R7,IR1
                LDI     R7,RC

                ADDI    1,R5
                CMPI    @LOG_SIZE,R5
                BLED    LOOP
                LDI     @DEST_ADDR,AR1
                LSH     -1,IR0
                LSH     1,R7

;
; Return to C environment.
;

                POP     DP              ; Restore C environment variables.
                POP     AR7
                POP     AR6
                POP     AR5
                POP     AR4
                POPF    R7
                POP     R7
                POPF    R6
                POP     R6
                POP     R5
                POP     R4
                POP     FP
                RETS

                .end

*
* No more.
*
*******************************************************************************
```

C. Software Source code

Software source code is provided by CD, because the quantity of source code is too large.

D. Reference Documents

D.1 Texas Instruments Documents

1) An A/D-D/A interface to the TMS320C40 global bus ---------- ---------- spra106.pdf
2) Designing with TMS320C40 Comm ports: part 1 ---------- spra213.pdf
3) Prototyping the TI TMS320C40 to the cypress vic068/vac068 VME interface ---------- spra105.pdf
4) TMS320C40 Boot loader selection ---------- spra208.pdf
5) TMS320 DSP Development support reference guide ---------- ---------- spru011f.pdf
6) TMS320C3x/c4x Assembly language tools user's guide ---------- spru035c.pdf
7) TMS320C3x/c4x Code generation tools getting started guide ---------- ---------- spru119b.pdf
8) TMS320C3x/c4x Optimizing C compiler user's guide ---------- ---------- spru034h.pdf
9) TMS320C4x Parallel runtime support library user's guide ---------- ---------- spru084a.pdf
10) TMS320C4x User's guide ---------- spru063b.pdf
11) TMS320C4x Parallel processing development system technical reference ---------- spru075a.pdf
12) TMS320C4x C source debugger user's guide ---------- spru054.pdf
13) TMS320C4x General-Purpose applications user's guide ---------- spru159.pdf

D.2 Etc Documents

1) Visual C++ Professional Programming ----------(YOUNGJUN PRESS)
2) Visual C++ Programming Bible ----------(YOUNGJUN PRESS)
3) PLX 9050 SDK Reference Guide ----------(PLX Corporation)
4) Interfacing TMS320C6x Series of DSP Processors to the PCI PUS ---------- (V3 Semiconductors)
5) A magazine MicroSoftware Jun. Jul. Aug. Sep. 1999 ----------(window driver programming)
6) VHDL ----------(SUNGANDANG PRESS)
7) ALTERA MAX+PLUS2 use a digital system plan ----------(BUKDOO PRESS)
8) ByteBlaster parallel Port Download Cable DataSheet ----------(ALTERA)

E. Index of Firmware Library

int c3xtimer();

int c3xtimer_reset();

int c3xtimer_init();

int cad_memory_reset();

int cad_init();

int cad_start();

int cad_end();

int cget_ad();

int cget_ad_ch0();

int cget_ad_ch1();

int cget_ad_ch2();

int cget_ad_ch3();

int cget_ad_flag();

int cget4_mail();

int cget4_mail_flag();

int cput_mail();

int Exponential_Window(int Window_Flag)

int ffft_rl();

int gain_ctrl_ch0();

int gain_ctrl_ch1();

int gain_ctrl_ch2();

int gain_ctrl_ch3();

int Hamming_Window( )

int Hanning_Window( )

int Rectangular_Window(int Window_Flag)

F. Index of Software Library

```
int WINAPI open_vxd();
void WINAPI close_vxd();
void WINAPI set_hwnd(HWND hWnd);
int WINAPI pci_init();
int WINAPI pci_close();
int WINAPI put2_mail(int dat, int i);
int WINAPI get_mail(int box);
int WINAPI reset();
int WINAPI install_interrupt();
int WINAPI uninstall_interrupt();
int WINAPI install_timer();
int WINAPI reset_timer();
int WINAPI download(char* filename);
int WINAPI send_cmd(unsigned int ui);
int WINAPI set_time_count(int time_count);
int WINAPI set_gain_ctrl(int channel, int value);
double* WINAPI get_time_data_addr(int channel);
double* WINAPI get_spectral_data_addr(int channel);
int WINAPI begin_daq();
int WINAPI end_daq();
int WINAPI get_vxd_buff_pos();
void WINAPI begin_process();
void WINAPI close_process();
int WINAPI get_cvt(float* data, int size=MAX_CHAN);
double WINAPI get_sts_channel(int channel);
int WINAPI save_file(char *filename, int channel);
```