ADLINK TECHNOLOGY INC.

GEME-2000/3000 Series

General Embedded Machine Engine

User's Manual

Manual Rev.	2.01
Revision Date:	October 19, 2005
Part No:	50-1D002-2020

Recycled Paper

Advance Technologies; Automate the World.

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

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

Trademarks

GEME® is a registered trademark of ADLINK Technology Inc. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting Service from ADLINK

Customer Satisfaction is top priority for ADLINK Technology Inc. Please contact us should you require any service or assistance.

ADLINK TECHNOLOGY INC.

Web Site:	http://www.adlinktech.com
Sales & Service:	Service@adlinktech.com
TEL:	+886-2-82265877
FAX:	+886-2-82265717
Address:	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan

Please email or FAX this completed service form for prompt and satisfactory service.

Company Information	
Company/Organization	
Contact Person	
E-mail Address	
Address	
Country	
TEL	FAX:
Web Site	
Product Information	
Product Model	
Environment	OS: M/B: CPU: Chipset: BIOS:

Please give a detailed description of the problem(s):

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Product Overview

The General Embedded Machine Engine (GEME) is a complete solution for Factory Automation (FA) and Machine Automation (MA) system integrators.

GEME is a rugged and compact chassis that supports an embedded SBC and power supply unit with optional storage peripherals, such as CompactFlash cards or a 2.5" HDD. Software compatibility issues can also be avoided through its built-in embedded software.

Although GEME is highly integrated, it can be further expanded with one PMC and three PC/104 extension modules, allowing GEME to additionally support motion, vision, DIO, communications, and High Speed Link applications.

With both hardware and software integrated in a single package, GEME is optimized for performance and reliability.

Important features of GEME:

- Low power consumption, fanless CPU applied for embedded applications
- Versatile functionalities: motion, vision, DIO, communications, High Speed Link
- Expandable enclosure design for one PMC and up to three PC104 modules
- Compact and rugged system design with wall-mounting kit
- OS support: Windows CE, Windows XP Embedded, and Linux

1.2 Unpacking Checklist

Check the shipping carton for any damage. If the shipping carton and contents are damaged, notify the dealer for a replacement. Retain the shipping carton and packing materials for inspection by the dealer. Please obtain authorization before returning any product to ADLINK.

Check the following items are included in the package, if there are any items missing, please contact your dealer:

Product	Included Items
GEME-2000/ 3000 Series	All-in-one support CD-ROM (software & manuals)
	Wall mounting kit: Wall mounting bracket (x4) M4 8mmScrew (x8)
	Power cord (L=1.8m, Please specify the country where this will be used in the ordering process) (for AC type only)
	PS/2 Y cable
	IDE flat cable (40 pin, L=57cm)
	FDD flat cable (34pin, L=52cm)
	4-pin DC output harness for external drive (43cm)
GEME-V2000/ S2000/V3000/ S3000/X3000 Series	All-in-one support CD-ROM (software & manuals)
	Wall mounting kit: Wall mounting bracket (x4) M4 8mmScrew (x8)
	Power cord (L=1.8m, Please specify the country where this will be used in the ordering process) (for AC type only)
	PS/2 Y cable

Table 1-1: Items checklist

Note: The packaging of the GEME OEM version with non-standard configuration, functionality, or package may vary according to different configuration requests.

CAUTION: The board fitted inside the GEME system must be protected from static discharge and physical shock. Never remove any of the socketed parts except at a static-free workstation. Use the anti-static bag shipped with the product to handle the board. Wear a wrist strap grounded through one of the system's ESD Ground jacks when servicing system components.

1.3 GEME Family

The GEME family can be classified by CPU performance into the 2000 and 3000 series. It can be further distinguished by the availability of vision functionality. Extension modules can be ordered to provide other functions not standard on the base unit. Please refer to the following table for available base units in the GEME family.

	CPU	Vision
GEME-2000, GEME-S2000	Celeron 650	X
GEME-V2000	Celeron 650	Y
GEME-3000, GEME-S3000	Pentium III 800	X
GEME-V3000, GEME-X3000	Pentium III 800	Y
GEME-VM3000*	Pentium III 800	Y

Table 1-2: GEME Family

Note: Hardware identical to V4000. MPEG4 software video compression capabilities for security and remote video surveillance applications.

1.4 Specifications

		GEME-2000, GEME-S2000	GEME-V2000	GEME-3000, GEME-S3000	GEME-V3000, GEME-X3000
SBC	SBC Model No.	EBC-C200	EBC-C200V	EBC-P300	EBC-P300V
	CPU	Ultra Low Voltage Celeron 650MHz		Low Voltage Pentium III 800MHz	
	Cache	256 KB on-die Advanced Transfer Cache (ATC)		512 KB on-die Advanced Transfer Cache (ATC)	
	System Memory	One 144-pin SODIMM sockets, accepts up to 256 MB un-buffered SDRAM			
	Chipset	Intel 815E AGP chipset			
		82815E Graphics and Memory Controller Hub (GMCH)			
		82801BA I/O Controller Hub 2 (ICH2)			
	VGA	On-board VGA controller built-in AGP (3D hyper pipelined architecture)			
		Up to 1600 x 1200 in 8-bit color at 85 Hz refresh rate			
		Video memory sharing from main memory with Intel Dynamic Video Memory Technology (DVMT)			
		Memory size is controlled by device driver from 1MB up to 11MB			
	BIOS	Award BIOS, support PnP			
	Video Capture	No	Conexant Fusion878A Video decoder processor 4-CH NTSC/PAL input	No	Conexant Fusion878A Video decoder processor 4-CH NTSC/PAL input, one decoder processor for GEME-V3000, four decoder processor for GEME-X3000
	USB	Two USB ports, USB 1.1 compliant Two extra USB ports, USB 2.0 compliant (on GEME-S2000/S3000)			
	IEEE 1394	Texas Instruments TSB43AB23 1394a-2000 OHCI PHY/link-layer controller Three IEEE-1394 ports (two external , one internal)			
	Ethernet	Intel 82562EM 10BaseT/100BaseTx			
	Enhanced IDE	Bus Master IDE controller, EIDE interfaces for up to two devices, support PIO Mode 3/4 or Ultra DMA/100 IDE devices, including Hard Disk Drive, ATAPI CD-ROM, LS120, and ZIP drives.			
	CompactFlash	50 pin socket for CompactFlash Type I/II One extra 50 pin socket for CompactFlash Type I/I (on GEME-S2000/S3000)			
	Super I/O Chipset	Winbond W83627HF			
	PCI to ISA Bridge	Integrated Technology IT8888F PCI to ISA Bridge			
	Hardware Monitoring	Built-in Winbond W83627HF, monitoring CPU temperature, voltage and battery,+3.3V,+5V,+12V voltage			
	COM Ports	COM1/ COM2: 16550 UART compatible ports with RS-232 interface			
	Parallel Port	One high-speed parallel port, SPP/EPP/ECP mode			
	Keyboard/Mouse	Combed PS/2 type mini-DIN connectors			
	Floppy Interface	Supports two floppy drives (360kB, 720kB, 1.2MB, 1.44MB, 2.88MB), 34-pin header on-board. Front panel 34-pin connector available for external drive on GEME-2000/3000.			
	PMC Interface	1 on-board 32-bit 33Mhz PMC module socket for functionally expansion			
	PC104 Interface	16-bit, PC/104 interface for functionally expansion			
	AGP Module Interface	AGP 1.5V interface reserved on internal MiniPCI connector			
	Watchdog Timer	Time-out timing select 0-255 seconds or 0-255 minutes			
	Dimensions	129mm x 167.5mm			

Table 1-3: GEME 2000/3000 Specifications

		GEME-2000, GEME-S2000	GEME-V2000	GEME-3000, GEME-S3000	GEME-V3000, GEME-X3000
System	Power Supply(optional)	Universal input AC 100 VAC to 220 VAC, Max. output : +5V 11.5A, +12V 3A, -12V 0.5A			
		DC input: 10VDC to 30VDC, Max. input current: 13A at 10VDC, Max. output: +5V 10A, +12V 1.5A, -12V 0.3A			
	Operating Temp.	-10°-55°C	-10°-50°C	-10°-55°C	-10°-50°C
	Humidity	0%-90%			
	Dimensions	183x140x95.36 mm (wall mount kit not included) [16.84 mm(H) for each extension kit]			
	Power Consumption	With 256 MB SDRAM +5V 4.5A, +12V 300mA		With 256MB SDRAM +5V 6.5A, +12V 300mA	
		Test conditions: (1) CPU 100% loading (2) No HDD, CD ROM, extension module			
	Power Output	+5V Max. 1A, +12V Max. 1A	None	+5V Max. 1A, +12V Max. 1A	None
	Storage	Internal : One 44-pin IDE Disk on Chip (DOC) interface	Internal : one 44-pin IDE	Internal : One 44-pin IDE Disk on Chip (DOC) interface	Internal : one 44-pin IDE
		External : One 40-pin IDE One 34 Pin FDD	External : none	External : One 40-pin IDE One 34 Pin FDD	External : none
	GPIO (TTL)	none	one digital input, one digital output, one programmable trigger output	none	one digital input, one digital output, one programmable trigger output
	Operating System	Windows CE, Windows XP Embedded, Linux			
	Random Vibration	Operating: 5-100Hz, 0.00142 g2/Hz; 100-500Hz, -6dB/Octave, 0.5Grms, 3axes, 30 minutes/axis Non-operating: 5-100Hz , 0.02g2/Hz ; 100-500Hz , -6dB/Octave, 1.88Grms, 3 axes,1hr/axis(IEC 68-2-64)			

Table 1-3: GEME 2000/3000 Specifications

WARNING: Always disconnect the power cord from the chassis when working on it. Do not connect the power cord while the power switch is on. A sudden rush of power can damage sensitive electronic components. Only authorized and experienced electronics personnel should open the chassis.

CAUTION: Always ground yourself to remove any static electric charge before touching GEME. Modern electronic devices are very sensitive to static electric charges. Use a grounding wrist strap at all times. Place all electronic components on a static-dissipative surface or in a static-shielded bag.

2 Base Unit

This chapter will familiarize the user with available GEME interfaces and connections before getting started.

2.1 Dimensions

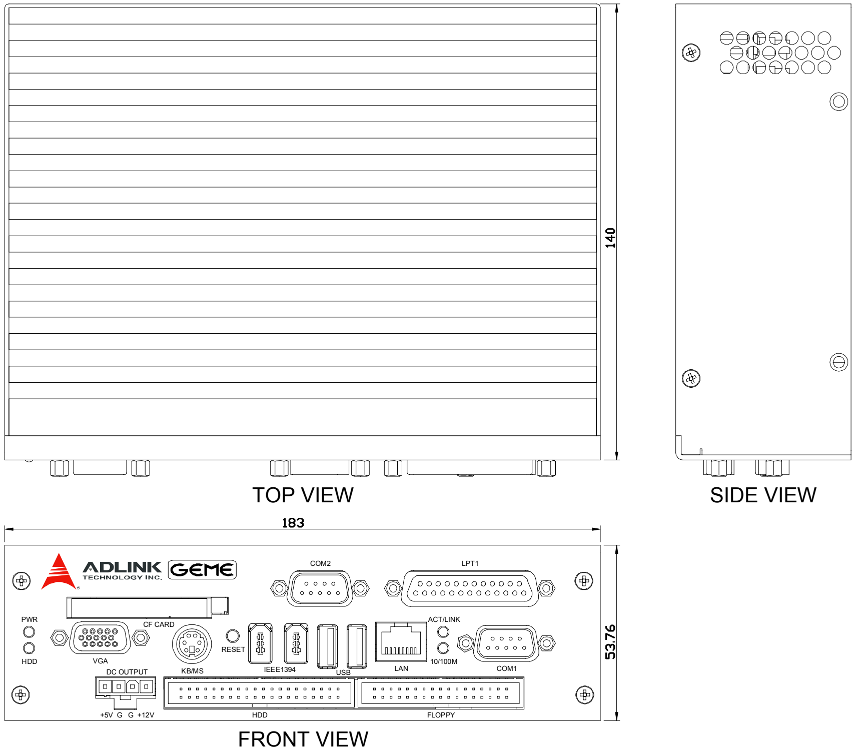


Figure 2-1: GEME-2000/3000 Base Unit Dimensions

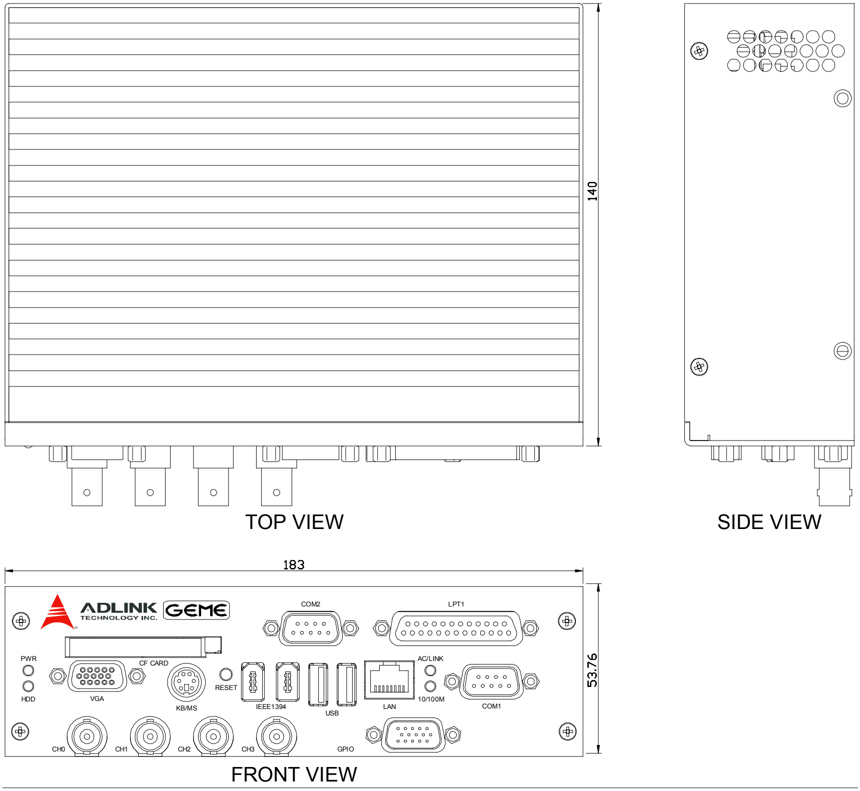


Figure 2-2: GEME-V2000/V3000/X3000 Base Unit Dimensions

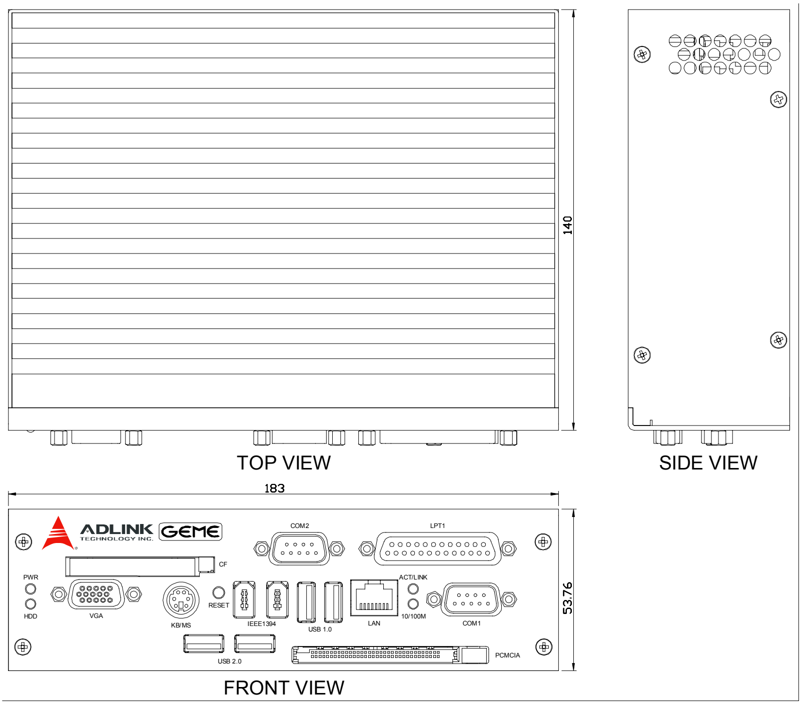


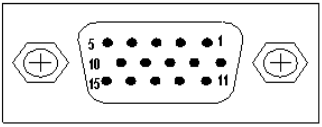
Figure 2-3: GEME-S2000/S3000 Base Unit Dimensions

2.2 Base Unit Connector Pin Assignments

Detailed descriptions and pin-outs for each connector are given in the following sections.

VGA Connector

GEME provides a VGA controller for a high resolution VGA interface. It supports VGA and VESA, up to 1280 x 1024 at 24bits, and video memory sharing from main memory with Intel Dynamic Video Memory Technology (DVMT). Memory size is controlled by the device driver from 1MB up to 11MB.



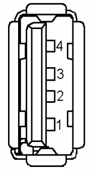
Signal Name	Pin	Pin	Signal Name
Red	1	2	Green
Blue	3	4	N.C.
GND	5	6	GND
GND	7	8	GND
+5V	9	10	GND
N.C.	11	12	DDCDAT
HSYNC	13	14	VSYNC
DDCCLK	15		

Table 2-1: VGA Connector

USB Connector

The USB connector can be used for connecting any device that conforms to the USB 1.1 specification. Many recent digital devices conform to this standard. The USB interface supports Plug and Play and hot-swap, which recognizes devices automatically and enables the user to connect or disconnect a device whenever needed to, without powering down the computer.

- ▶ GEME provides two USB interface connectors
- ▶ Plug and Play and hot-swap for up to 127 external devices. The USB compliant with USB Specification Rev. 1.1, individual over-current protection.

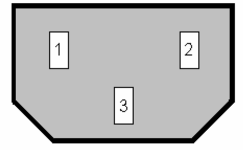


Pin	Signal
1	VCC
2	USB-
3	USB+
4	Ground

Table 2-2: USB Connector

AC Input Connector

GEME comes with an AC inlet connector that carries 100~240 VAC external power input, and features reversed wiring protection.

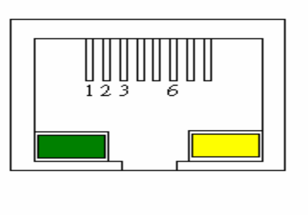


PIN	SIGNAL
1	Neutral
2	Line
3	Earth Ground

Table 2-3: AC Input Connector

Ethernet (RJ-45) Connector

GEME is equipped with Intel Ethernet LAN controller that is fully compliant with IEEE 802.3u 10/100Base-T CSMA/CD standards. The Ethernet port provides a standard RJ-45 jack onboard, and LED indicators on the front side to show its speed (Yellow LED) and Active/Link (Green LED) status.



Pin	Signal Name
1	TD+
2	TD-
3	RD+
4	NC
5	NC
6	RD-
7	NC
8	NC

Table 2-4: Ethernet (RJ-45) Connector

LAN status LED

LED Color	Status	Function
Yellow (Speed status)	ON	100Mbps
	OFF	10Mbps
Green (Link status)	ON	Link
	OFF	Link off
	Blinking	Data transfer in progress

Table 2-5: LAN status LED

COM1/COM2

GEME offers two serial communications interface ports: COM 1 and COM 2.

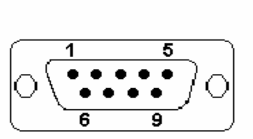
IRQ and Address Setting

The IRQ and I/O address range are both assigned by BIOS. The table below describes COM1/COM2 default settings:

COM Port	Mode	Bass address	IRQ
COM 1	RS-232	3F8h	IRQ4
COM 2	RS-232	2F8h	IRQ3

Table 2-6: IRQ and Address Setting

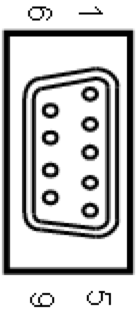
COM1/COM2 Pin Assignment



Pin	RS-232
1	DCD, Data carrier detect
2	RXD, Receive data
3	TXD, Transmit data
4	DTR, Data terminal ready
5	GND, ground
6	DSR, Data set ready
7	RTS, Request to send
8	CTS, Clear to send
9	RI, Ring indicator

Table 2-7: COM1 Pin Assignment

Note: COM 2 can also support RS422 or RS485 (without Auto direction function) by adjusting the jumper. Please refer to the EBC board manual for jumper configuration. This feature is available on request when placing the order.

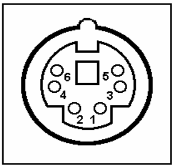


Pin	RS422	RS485
1	TX-	D-
2	TX+	D+
3	RX+	--
4	RX-	--
5	GND	GND
6	--	--
7	--	--
8	--	--
9	--	--

Table 2-8: COM2 Pin Assignment

Integrated PS/2 KBD/MS connector

GEME has a proprietary interface for PS/2 keyboard and mouse connections. A 6-pin mini-DIN connector is located on the rear panel of GEME. A proprietary ADLINK Y-cable is used to convert the 6-pin mini-DIN connector to two 6-pin mini-DIN connectors for the PS/2 keyboard and PS/2 mouse connections. The power provided to the keyboard and mouse is protected by a polyswitch rated at 1.1A.

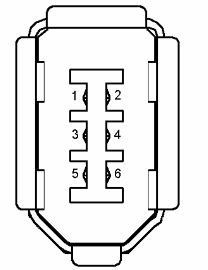


Pin	Signal	Function
1	KBDAT	Keyboard Data
2	MSDAT	Mouse Data
3	GND	Ground
4	KBMS5V	Power
5	KBCLK	Keyboard Clock
6	MSCLK	Mouse Clock

Table 2-9: Integrated PS/2 KBD/MS connector

IEEE1394 Connector

GEME comes with two IEEE 1394 interfaces, which are fully Plug & Play compliant and hot swappable. GEME's IEEE 1394 interface fully supports the IEEE 1394-1995 standard for high-performance serial bus and the IEEE 1394a-2000 supplement. Full IEEE 1394a-2000 support includes: connection debounce, arbitrated short reset, multi speed concatenation, arbitration acceleration, fly-by concatenation, and port disable/suspend/ resume. GEME has two IEEE 1394a-2000 fully compliant cable ports with transfer rates of 100/200/400 megabits per second (Mbits/s).



PIN	SIGNAL
1	+12V
2	GND
3	TPB0-
4	TPB0+
5	TPA0-
6	TPA0+

Table 2-10: IEEE1394 Connector

Compact Flash Connector

GEME’s standard CompactFlash (CF) socket has an ATA interface that is fully compatible with an IDE HDD and supports both type-I and type-II CF cards. The CF socket is on the Secondary IDE port.

Parallel Port Connector

Parallel I/O interface signals are routed to a DB25 socket on the front panel. This port supports full IEEE-1284 capability and provides a basic printer interface that supports EPP and ECP enhanced port modes.

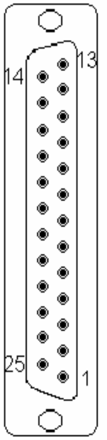
IRQ and Address Setting

The IRQ, I/O address range and mode are all assigned by BIOS. The following table outlines the parallel port default settings:

Parallel Port Mode	Bass address	IRQ
SPP	378h	IRQ7

Table 2-11: IRQ and Address Setting

Printer Port Connector



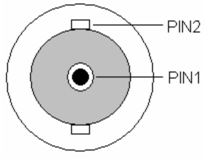
Signal Name	Pin	Pin	Signal Name
Line printer strobe	1	14	AutoFeed
PD0, parallel data 0	2	15	Error
PD1, parallel data 1	3	16	Initialize
PD2, parallel data 2	4	17	Select
PD3, parallel data 3	5	18	Ground
PD4, parallel data 4	6	19	Ground
PD5, parallel data 5	7	20	Ground
PD6, parallel data 6	8	21	Ground
PD7, parallel data 7	9	22	Ground
ACK, acknowledge	10	23	Ground
Busy	11	24	Ground
Paper empty	12	25	Ground
Select	13	N/A	N/A

Table 2-12: Printer Port Connector

Video Capture BNC Connector (GEME-V2000/V3000/X3000)

GEME accepts 4 channels standard composite color (PAL, NTSC) or monochrome video formats (CCIR, EIA).

Video resolution is programmable including the square-pixel (640 x 480 or 768 x 576) and the broadcast resolution.

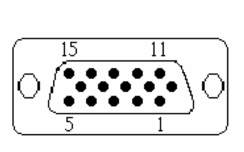


PIN	SIGNAL
1	Video Signal
2	GND

Table 2-13: Video Capture BNC Connector

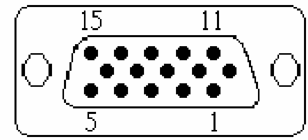
GPIO Connector (GEME-V2000/V3000/X3000)

GEME's I/O lines are TTL compatible and support single-input, single-output and single-software trigger lines.



PIN	SIGNAL	PIN	SIGNAL
1	Digital Input	9	NC
2	GND	10	GND
3	Digital Output	11	NC
4	GND	12	GND
5	Software Trigger	13	+12V
6	GND	14	
7	NC	15	
8	GND		

Table 2-14: GPIO Connector (GEME-V2000/V3000)



PIN	SIGNAL	PIN	SIGNAL
1	Digital Input 0	9	Digital Output 3
2	Digital Input 1	10	GND
3	Digital Input 2	11	GND
4	Digital Input 3	12	GND
5	GND	13	+5V
6	Digital Output 0	14	
7	Digital Output 1	15	
8	Digital Output 2		

Table 2-15: GPIO connector (GEME-X3000)

The I/O lines are internally pulled up and have the following characteristics:

Voltage	MIN	MAX
Input high voltage (5μA)	2.0 V	5.25V
Input low voltage (-5μA)	0.0V	0.80V
Output high voltage (-1.0mA)	5.0V	-
Output low voltage (100.0mA)	-	0.5V

Table 2-16: I/O Line Voltage

Software trigger output :

Programmable trigger scale, from 60uS ~ 16mS

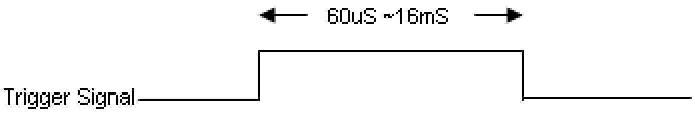
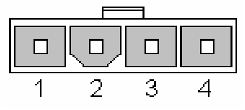


Figure 2-4: I/O Line Voltage

DC Power Output Connector

The DC power output is protected by a polyswitch rated at 1.1A.

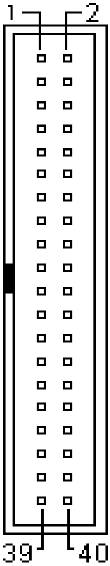


Pin	Signal
1	+12V
2	GND
3	GND
4	+5V

Table 2-17: DC Power Output Connector

IDE Interface Connector

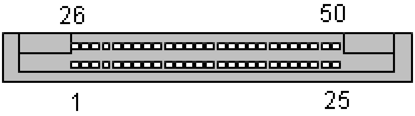
GEME has primary IDE interfaces for up to two devices, supporting PIO Mode 3/4 or Ultra DMA/100 IDE devices, including Hard Disk Drive, ATAPI CD-ROM, LS120, and ZIP drives.



Signal Name	Pin	Pin	Signal Name
Reset IDE	1	2	Ground
Host data 7	3	4	Host data 8
Host data 6	5	6	Host data 9
Host data 5	7	8	Host data 10
Host data 4	9	10	Host data 11
Host data 3	11	12	Host data 12
Host data 2	13	14	Host data 13
Host data 1	15	16	Host data 14
Host data 0	17	18	Host data 15
Ground	19	20	+5V
DRQ0 / DRQ1	21	22	Ground
Host IOW	23	24	Ground
Host IOR	25	26	Ground
IOCHRDY	27	28	Host ALE
DACK0 / DACK1	29	30	Ground
IRQ14 / IRQ 15	31	32	IOCS16#
Address 1	33	34	No connect
Address 0	35	36	Address 2
Chip select 0	37	38	Chip select 1
Activity	39	40	Ground

Table 2-18: IDE Interface Connector

GEME has secondary IDE interfaces on the front panel, supporting compact flash Type I/II.

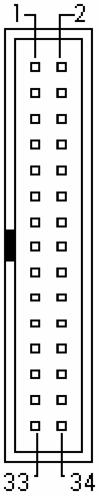


Pin	Signal Name	Pin	Signal Name
1	GND	26	CD1-
2	DATA3	27	DATA11
3	DATA4	28	DATA12
4	DATA5	29	DATA13
5	DATA6	30	DATA14
6	DATA7	31	DATA15
7	CE1#	32	CE2#
8	A10	33	VS1
9	OE#	34	IOR#
10	A9	35	IOW#
11	A8	36	WE#
12	A7	37	READY#
13	BVCC	38	BVCC
14	A6	39	NC
15	A5	40	VS2
16	A4	41	RESET
17	A3	42	WAIT#
18	A2	43	INPACK#
19	A1	44	REG#
20	A0	45	BVD2
21	DATA0	46	BVD1
22	DATA1	47	DATA8
23	DATA2	48	DATA9
24	WP	49	DATA10
25	CD2#	50	GND

Table 2-19: Secondary IDE

Floppy Interface Connector

GEME supports up to two floppy drives (360KB, 720KB, 1.2MB, 1.44MB, 2.88MB)



Signal Name	Pin	Pin	Signal Name
Ground	1	2	Drive density selection
Ground	3	4	No connect
No connect	5	6	Drive density selection
Ground	7	8	Index
Ground	9	10	Motor enable 0
Ground	11	12	Drive select 1
Ground	13	14	Drive select 0
Ground	15	16	Motor enable 1
Ground	17	18	Direction
Ground	19	20	Step
Ground	21	22	Write data
Ground	23	24	Write gate
Ground	25	26	Track 00
Ground	27	28	Write protect
Ground	29	30	Read data
Ground	31	32	Side 1 select
Ground	33	34	Diskette change

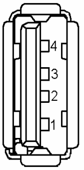
Table 2-20: Floppy Interface Connector

USB 2.0 Connector (GEME-S2000/S3000)

The USB connectors can be used for connecting any device that conforms to the USB 2.0 specification. Many recent digital devices conform to this standard. The USB interface supports Plug and Play and Hot Swapping, which recognizes devices automatically and enables you to connect or disconnect a device whenever you want without powering down the computer.

- GEME-S series provides two connectors for USB 2.0 interfaces

- Plug & Play and hot swapping for up to 127 external devices is supported.
- Compliant with USB Specification Rev. 2.0; individual over-current protection.



Pin	Signal
1	VCC
2	USB-
3	USB+
4	Ground

Table 2-21: USB 2.0 Connector

PCMCIA Interface (GEME-S2000/S3000)

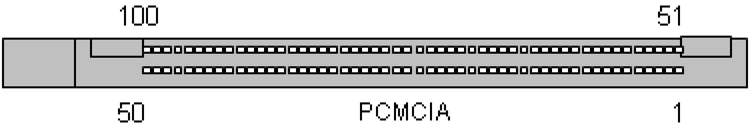
One TYPEI/II PCMCIA slot, complies with PC Card Standard 8.1.

PIN	Signal Name	PIN	Signal Name	PIN	Signal Name	PIN	Signal Name
1	GND	26	GND	51	GND	76	GND
2	DATA3	27	VPP	52	CD1	77	VPP
3	DATA4	28	A16	53	DATA11	78	A22
4	GND	29	GND	54	GND	79	GND
5	DATA5	30	A15	55	DATA12	80	A23
6	DATA6	31	A12	56	DATA13	81	A24
7	GND	32	GND	57	GND	82	GND
8	DATA7	33	A7	58	DATA14	83	A25
9	CE1#	34	A6	59	DATA15	84	VS2
10	GND	35	GND	60	GND	85	GND
11	A10	36	A5	61	CE2#	86	RESET
12	OE	37	A4	62	VS1	87	WAIT#
13	GND	38	GND	63	GND	88	GND
14	A11	39	A3	64	IORD#	89	INPACK#
15	A9	40	A2	65	IOWR#	90	REG

Table 2-22: PCMCIA Interface

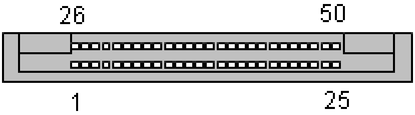
PIN	Signal Name	PIN	Signal Name	PIN	Signal Name	PIN	Signal Name
16	GND	41	GND	66	GND	91	GND
17	A8	42	A1	67	A17	92	ABVD2
18	A13	43	A0	68	A18	93	ABVD1
19	GND	44	GND	69	GND	94	GND
20	A14	45	DATA0	70	A19	95	DATA8
21	WE#	46	DATA1	71	A20	96	DATA9
22	GND	47	GND	72	GND	97	GND
23	RDY	48	DATA2	73	A21	98	DATA10
24	VCC	49	WP	74	VCC	99	CD2
25	GND	50	GND	75	GND	100	GND

Table 2-22: PCMCIA Interface



Second CampactFlash Interface (GEME-S2000/S3000)

One TYPEI/II compact flush slot on the GEME's side, support compact flash hot swap. The system OS can't boot from this compact flash interface.



Pin	Signal Name	Pin	Signal Name
1	GND	26	CD1-
2	DATA3	27	DATA11
3	DATA4	28	DATA12
4	DATA5	29	DATA13
5	DATA6	30	DATA14
6	DATA7	31	DATA15
7	CE1#	32	CE2#
8	A10	33	VS1
9	OE#	34	IOR#
10	A9	35	IOW#
11	A8	36	WE#
12	A7	37	READY#
13	BVCC	38	BVCC
14	A6	39	NC
15	A5	40	VS2
16	A4	41	RESET
17	A3	42	WAIT#
18	A2	43	INPACK#
19	A1	44	REG#
20	A0	45	BVD2
21	DATA0	46	BVD1
22	DATA1	47	DATA8
23	DATA2	48	DATA9
24	WP	49	DATA10
25	CD2#	50	GND

Table 2-23: Secondary IDE

3 Power Supply Unit

The entire GEME series can support either AC or DC power supplies per application requirements. The power supply unit is internally integrated into GEME system.

3.1 Dimensions

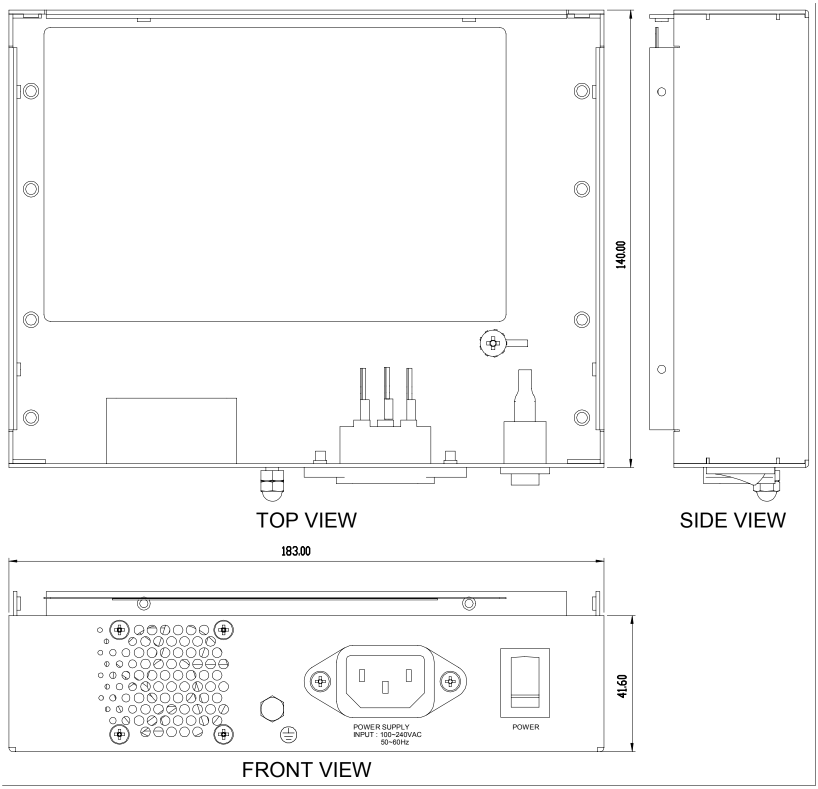


Figure 3-1: GEME AC type power supply unit dimensions

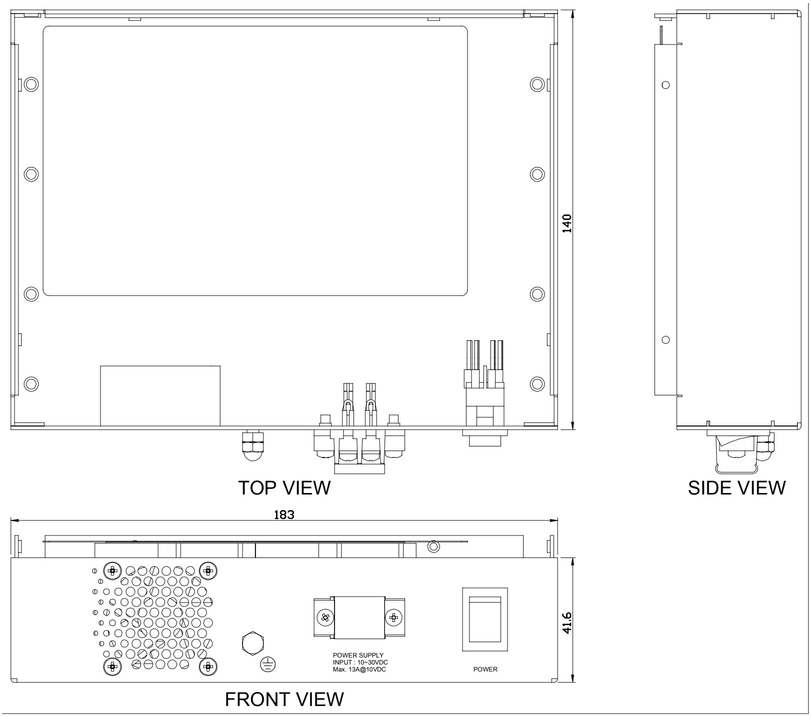


Figure 3-2: GEME DC type power supply unit dimensions

3.2 Specifications

AC Power Supply

The AC power supply is a triple output 110W switching power supply, which is designed to meet Harmonics EN61000-3-2.

Input Specifications

Input voltage	The range of input voltage is from 91 to 264VAC. The nominal voltage is 115VAC 60Hz and 230VAC 50Hz.
Input frequency	The range of input frequency is from 47Hz to 63Hz
Input current	The maximum input current is 2A at 115VAC 1A at 230VAC
Inrush current	The inrush current will not exceed 30A at 115VAC input or 60A at 230VAC input, cold start, 25ºC

Table 3-1: AC Power Supply - Input Specifications

Output Specifications

Output	Rated load	Peak load
+5V	11.5A	15A
+12V	3A	5A
-12V	0.5A	0.5A

Table 3-2: AC Power Supply - Output Specifications

International Standards

Safety standards	**UL 60950** CSA 22.2 NO.234 **EN 60 950**
EMI standards	FCC docket 20780 curve “B” EN 55022“B” EN 61000-3-2
EMS standards	▶ EN61000-4-2: 6KV contact discharge, 8KV air discharge Criteria A ▶ EN61000-4-3: 10V/m Criteria A ▶ EN61000-4-4: 2KV Criteria A

Table 3-3: AC Power Supply - International Standards

DC Power Supply

The DC power supply is a 72W triple-output switching power supply, specially designed for microprocessor-based applications; DC input from 10V to 30V; enclosed type.

Input Specifications

Input voltage	This power supply can operate continuously from +10VDC to +30VDC, normal line is +24VDC
Input current	The maximum input current is 13A at 10VDC
Inrush current	The maximum inrush current will not exceed 25A at 12VDC input from a cold start, with the exclusion of EMI capacitors

Table 3-4: DC Power Supply - Input Specifications

Output Specifications

Output	Rated load	Peak load
+5V	10A	14A
+12V	1.5A	3A
- 12V	0.3A	

Table 3-5: DC Power Supply - Output Specifications

International Standards

Safety standards	UL 1950 CSA 22.2 No. 234 VDE EN 60950
EMI standards	FCC docket 20780 curve “B” EN55022 class “B”
EMS standards	IEC-801-2 8KV air discharge IEC-801-3 3V/M IEC-801-4 2KV

Table 3-6: DC Power Supply - International Standards

4 Getting Started

4.1 Storage Settings

HDD / Compact Flash Card

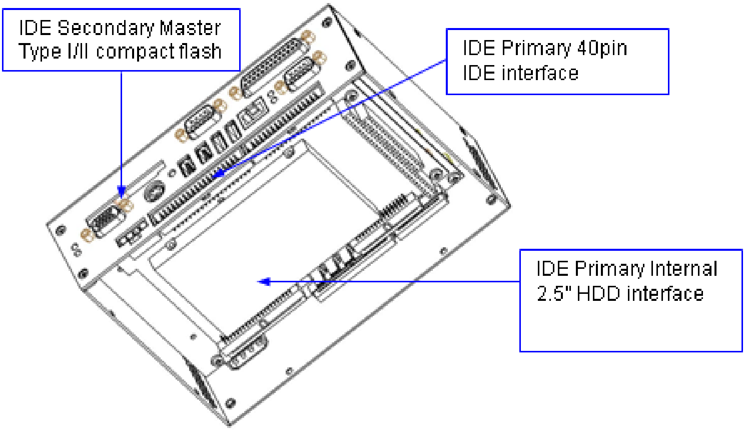


Figure 4-1: HDD/Compact Flash Card

Note:	1. The diagram above is intended for describing the IDE interfaces only, not for disassembly. 2. The IDE primary 40 pin IDE interface is for customers' external use. 3. The IDE primary internal 2.5" HDD interface is designed for internal use, hence the device will be installed by ADLINK according to customer's request.

4.2 IDE Boot Sequence Settings

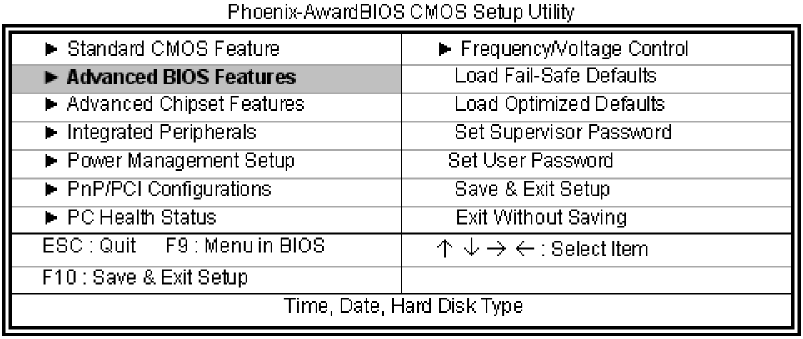
	IDE device 1			IDE device 2			IDE device 3
IDE Primary Master	HDD-0	--	--	HDD-0	HDD-0	--	HDD-0
IDE Primary Slave	--	HDD-0	--	HDD-1	--	HDD-0	HDD-1
IDE Secondary Master	--	--	HDD-0	--	HDD-1	HDD-1	HDD-2

Table 4-1: IDE Boot Sequence Settings

The Phoenix-Award BIOS provides a Setup utility program for specifying system configuration and settings. The Setup utility is stored in BIOS ROM. When the system is powered up, BIOS is activated. Press the <Del> key immediately to enter the Setup utility. If there is a delay in pressing the <Del> key after BIOS is activated, POST (Power On Self Test) will continue with its test routines, thus preventing the user from entering Setup. Should the user still wish to enter into Setup, restart the system by pressing the “Reset” button or simultaneously pressing the <Ctrl>, <Alt>, and <Delete> keys. The system can also be restarted by switching the system off and back on again. Upon power up, the following message will appear on the screen:

```
Press <DEL> to Enter Setup
```

In the Setup utility program, the user can make changes by pressing the arrow keys to highlight items, <PgUp> and <PgDn> keys to change entries, <Enter> to select, <F1> for help, and <Esc> to quit. When the user enters the Setup utility, the Main Menu screen will appear on the screen. The Main Menu allows the user to select from various setup functions and exit choices.

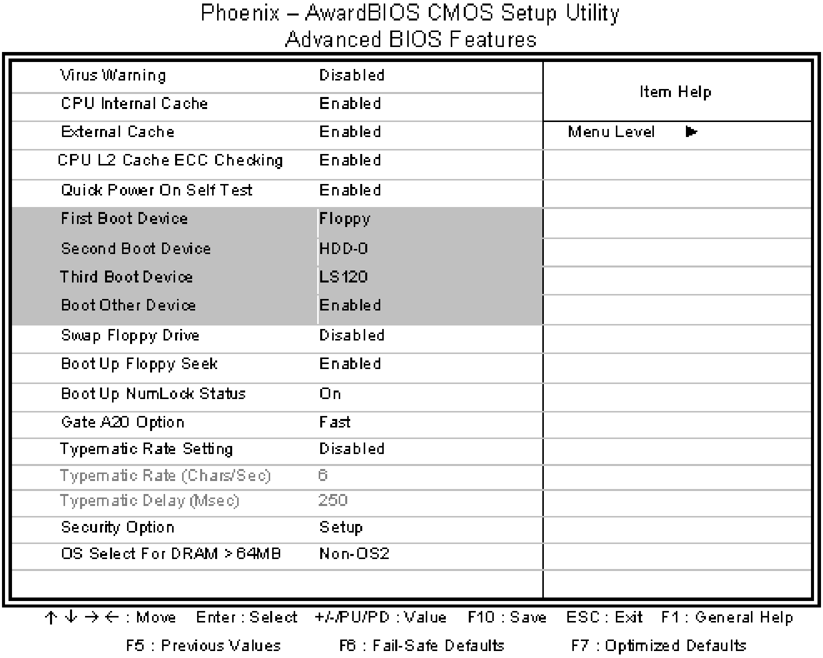


Phoenix-AwardBIOS CMOS Setup Utility

► Standard CMOS Feature	► Frequency/Voltage Control
► Advanced BIOS Features	Load Fail-Safe Defaults
► Advanced Chipset Features	Load Optimized Defaults
► Integrated Peripherals	Set Supervisor Password
► Power Management Setup	Set User Password
► PnP/PCI Configurations	Save & Exit Setup
► PC Health Status	Exit Without Saving
ESC : Quit F9 : Menu in BIOS	↑ ↓ → ← : Select Item
F10 : Save & Exit Setup	

Time, Date, Hard Disk Type

For IDE boot sequence setting, select **Advanced BIOS Features:**



Phoenix – AwardBIOS CMOS Setup Utility
Advanced BIOS Features

Virus Warning	Disabled	Item Help
CPU Internal Cache	Enabled	
External Cache	Enabled	Menu Level ►
CPU L2 Cache ECC Checking	Enabled	
Quick Power On Self Test	Enabled	
First Boot Device	Floppy	
Second Boot Device	HDD-0	
Third Boot Device	LS120	
Boot Other Device	Enabled	
Swap Floppy Drive	Disabled	
Boot Up Floppy Seek	Enabled	
Boot Up NumLock Status	On	
Gate A20 Option	Fast	
Typematic Rate Setting	Disabled	
Typematic Rate (Chars/Sec)	6	
Typematic Delay (Msec)	250	
Security Option	Setup	
OS Select For DRAM > 64MB	Non-OS2	

↑ ↓ → ← : Move Enter : Select +/-/PU/PD : Value F10 : Save ESC : Exit F1 : General Help
F5 : Previous Values F6 : Fail-Safe Defaults F7 : Optimized Defaults

First/Second/Third/Other Boot Device

BIOS attempts to load the operating system from the devices in the sequence selected in the following items. The settings are:

Disabled	Floppy	LS120
HDD-0	SCSI	CDROM
HDD-1	HDD-2	HDD-3
ZIP100	LAN	

Table 4-2: Boot Sequence options

Note: HDD-0, HDD-1, HDD-2 are for IDE device, and HDD-3 is not used.

4.3 IRQ Information

The IRQ and base address settings in the GEME system are set according to the "assembly order" of the PC104 cards in the GEME system. Please refer to the diagram below:

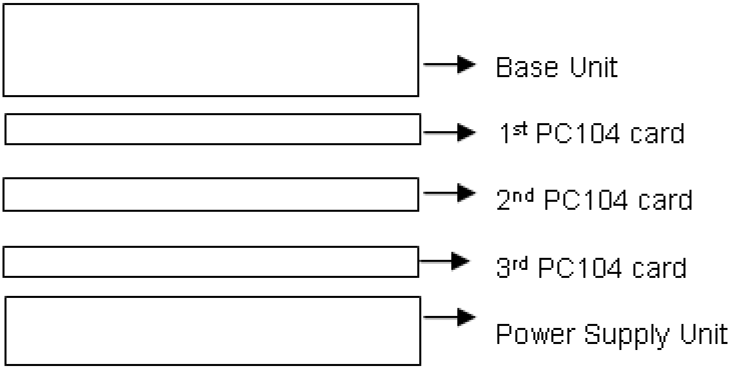


Table 4-3: IRQ Information

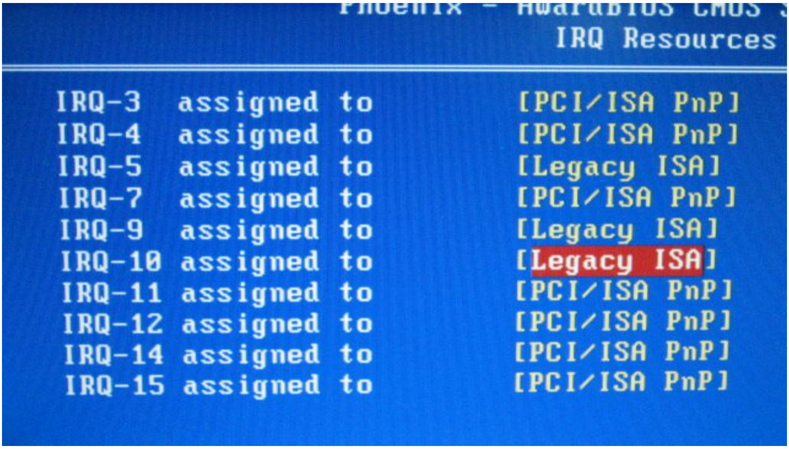
	1st PC104 card	2nd PC104 card	3rd PC104 card
I/O Address	300	200	280
IRQ No.	9	5	10
Memory Address	D0000	D4000	D8000

4.4 Software Settings

The GEME system software is installed before shipping according to the customers' configuration options. This section provides the necessary information for customers who need to rebuild their OS.

Step1:

Check IRQ resources setting in BIOS (Set the IRQ number according to the user's system configuration - please see section 4.3 for further information). For example, the user will be required to change the settings of IRQ 5, 9, and 10 as "Legacy ISA" in BIOS if there are three PC 104 cards in GEME. The remaining IRQ settings are set as "PCI/ISA PnP". When GEME is booted up, press the "DEL" key to enter the BIOS setting screen as follows.

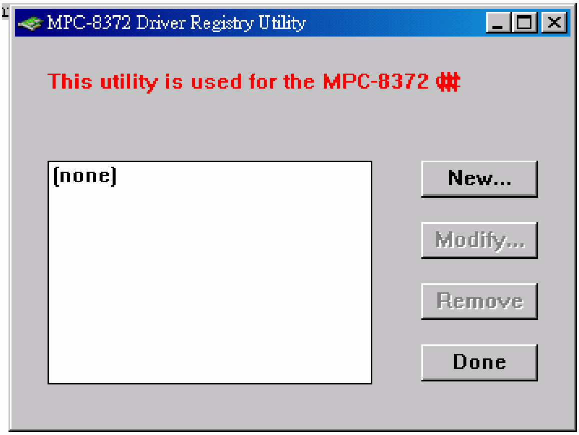


```
                 Phoenix - AwardBIOS CMOS
                                IRQ Resources

  IRQ-3   assigned to          [PCI/ISA PnP]
  IRQ-4   assigned to          [PCI/ISA PnP]
  IRQ-5   assigned to          [Legacy ISA]
  IRQ-7   assigned to          [PCI/ISA PnP]
  IRQ-9   assigned to          [Legacy ISA]
  IRQ-10  assigned to          [Legacy ISA]
  IRQ-11  assigned to          [PCI/ISA PnP]
  IRQ-12  assigned to          [PCI/ISA PnP]
  IRQ-14  assigned to          [PCI/ISA PnP]
  IRQ-15  assigned to          [PCI/ISA PnP]
```

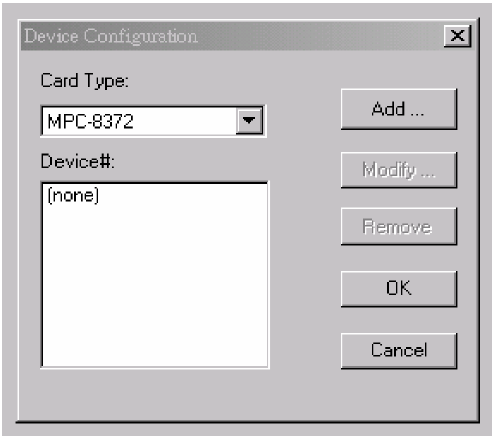
Step2:

The user will be required to register their PC104 card in the Windows system. For example, the user can find the "Registry Utility" from the MPC-8372 folder after installing the MPC-8372 driver. The screen is shown below. If there are other PC104 modules in GEME, the corresponding utility can also be found.



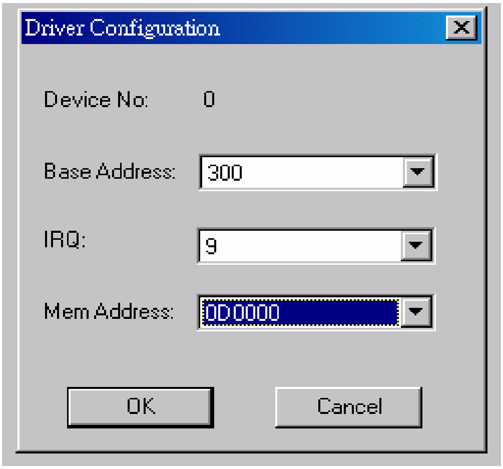
MPC-8372 Driver Registry Utility
This utility is used for the MPC-8372 ##
[none]
New...
Modify...
Remove
Done

After pressing “New” as shown in the screen above, the “Device Configuration” dialog box will appear. Enter the corresponding PC104 card information. For example, an additional MPC-8372 card can be added in the dialog if there is only one MPC-8372 in GEME. Press “New” to proceed.



Refer to the table in section 4.3 to select the “Base Address”, “IRQ”, and “Mem Address”. Follow the dialog boxes below if there is only one MPC-8372 card in the user’s GEME system.

Press “OK” to save this setting. These new settings will take effect once the Windows system is rebooted.



Driver Configuration

Device No: 0

Base Address: 300

IRQ: 9

Mem Address: 0D0000

4.5 Supported Software

The operating system is responsible for managing core tasks and resource allocation for the hardware. It not only affects future system execution time and efficiency, but also resource requirements during software development for the entire application.

When choosing an operating system, the following need to be considered: stability, real-time capability, multitasking, human-machine interface (or GUI), memory size, and total cost (including application development costs, licensing costs for multiple copies, software engineering manpower, maintenance costs, etc). GEME allows for maximum flexibility by being compatible with several operating systems.

Operation System Support

Windows 2000/XP

Windows2000/XP supports GEME's chipset drivers, allowing the user to install Windows 2000/XP themselves. ADLINK also provides OS pre-installation service for Windows 2000/XP on GEME (with a Windows 2000/XP license pre-purchased from ADLINK). As Windows 2000/XP requires a large amount of storage space, a hard drive is the best storage solution for GEMEs running Windows 2000/XP.

Linux

Most Linux distributions (e.g. RedHat, Suse, etc) also support GEME's chipset drivers, allowing users to install Linux themselves. As Linux also requires a large amount of storage space, a harddrive is the best storage solution for GEMEs running Linux.

Windows XP Embedded

Windows XP is a multitasking operating system known for its stability. As a result of its overwhelming popularity, human-machine interfaces, and many development tools, developing applications in Windows XP is comparatively simple. Microsoft carried over the advantages of Windows XP when releasing Embedded XP. The concept behind the design of Embedded XP is simply a modularized Windows XP. System developers only select the required Windows XP components and functions, then organize them to construct a XP Embedded OS. As a result of this architectural modularization, system integrators can readily reduce the storage space requirements of XP Embedded. The only factor determining storage space requirements is the number of function modules needed. Because XP Embedded is completely compatible with Windows XP, developers can compile controller software in the Windows XP environment, and transfer the code to Embedded XP for immediate use. System developers do not need to learn any new tools to use XP Embedded. Their skills in Windows XP can be directly transferred to XP Embedded, thus lowering software development costs. Another advantage is the cost of licensing Embedded XP is much lower than that of Windows XP.

Currently ADLINK provides the standard XP Embedded OS image for GEME (customers must pre-purchase an XP Embedded license from ADLINK). The standard XP Embedded OS image provided by ADLINK is approximately 200MB for the English version, and 400MB for the Chinese version. For this OS configuration, a compact flash card is the best storage device for GEME. The major functions inside the standard XP Embedded OS image are as follows:

- XP Embedded OS Kernel
- Drivers for GEME H/W and peripheral cards
- TCP/IP Networking
- TCP/IP with file sharing and client for Microsoft network
- Internet Explorer
- File Manager
- Language Support

The standard XP Embedded OS image can meet most application needs. If the customer has specific function requirements for XP Embedded, please contact ADLINK’s field application engineers (FAE) for further information.

Windows CE

As Windows CE is designed with embedded systems in mind, it requires less storage space than XP Embedded. Windows CE typically requires 64MB of storage space and is possible to reduce this amount if needed. An important feature of Windows CE is that it supports real-time functionality. Microsoft has tried to keep API naming conventions and the development process consistent between Windows CE and Windows XP. However, as Windows CE has an embedded architecture (as compared with Windows XP's desktop system concept), the software development process will have significant differences. Another factor to consider when deciding whether to use Windows CE is that licensing costs are much lower than XP Embedded.

Currently ADLINK offers the standard WinCE OS image for GEME (customer must purchase the WinCE license from ADLINK). The standard WinCE OS image requires about 25MB for the English version, and 30MB for the Chinese version. For this OS configuration, a compact flash card is the best storage device for GEME. The major functions included with the standard WinCE OS image are as follows.

- WinCE OS Kernel
- Drivers for GEME H/W and peripheral cards
- TCP/IP Networking
- Internet Explorer
- File Manager
- Language Support

The standard WinCE OS image can meet most application requirements. If the customer has special functional requirements for WinCE, please contact ADLINK's FAE for further information.

Driver Support

Driver support for GEME's peripheral cards under the different OS systems are outlined in the following tables. Drivers for these peripheral cards can be found in the ALL-IN-ONE CD (Automation).

Motion Cards

Module No.	Bus Interface	Description	Win2000, XP & eXP driver	Win CE driver	Linux driver
MPC-8164	PC104	4-axis pulse type motion	Ready	Ready	Ready
MPC-8372 / 66	PC104	12-axis / 6-axis SSCNET motion	Ready	Ready	Call for status

Table 4-4: Motion Cards

Communication Cards

Module No.	Bus Interface	Description	Win2000, XP & eXP driver	Win CE driver	Linux driver
PMC-3534G	PMC	4 port asynchronous serial comm.	Ready	Ready	Ready
PMC-3544G	PMC	4 port RS-422/485 isolated serial comm.	Ready	Ready	Ready
PMC-7841G	PMC	CAN bus communication card	Ready	Call for status	Call for status

Table 4-5: Communication Cards

HSL Card

Module No.	Bus Interface	Description	Win2000, XP & eXP driver	Win CE driver	Linux driver
PMC-7852G	PMC	HSL Serial I/O master card	Ready	Ready	Ready

Table 4-6: HSL Card

DIO Card

Module No.	Bus Interface	Description	Win2000, XP & XP driver	Win CE driver	Linux driver
MPC-7632/64	PC104	32/64 CH Digital I/O	Ready	Ready	Ready

Table 4-7: DIO Card

GEME Driver Installation List

	WinXP/2000	WinNT	WinXP embedded*(1)	WinCE	Linux*(2)
GEME add-on cards	Built-in OS	Users can install drivers with the ADLINK-ALL-IN-ONE (Automation) CD	Test & registration programs are built-in ADLINK standard image(C:\ADLINK)	Built-in ADLINK standard image	..
Vision	Built-in OS	Users can install drivers with the ADLINK-ALL-IN-ONE (Automation) CD	View Creator utility is built-in ADLINK standard image(C:\ADLINK\Angelo)	Built-in ADLINK standard image	..
MPEG4	Users can install drivers with the ADLINK-ALL-IN-ONE CD	Users can install drivers with the ADLINK-ALL-IN-ONE CD	*(1) Encode utility is built-in ADLINK standard image (C:\ADLINK\MPEG4)	X	X

Table 4-8: GEME Driver Installation List

Note(1): ADLINK will pre-install the hardware driver, utility, and runtime library on GEME. For developing program in the Host PC, the user must install the corresponding software package with ADLINK all-in-one CD.

Note(2): Please check with ADLINK FAE about WinCE & Linux

*** With the ADLINK ALL-IN-ONE (Automation) CD, users can install drivers for Windows 2000/XP systems. For XP Embedded systems, if the XP Embedded OS is built by ADLINK, ADLINK will pre-install the drivers for GEME's peripheral cards in the OS image. If users build their own XP Embedded OS image, they can also use the ADLINK-ALL-IN-ONE (Automation) CD for driver installation.

*** Currently the Linux drivers ADLINK provides for GEME's peripheral cards are based on Kernel 2.4.18 (RedHat 7.3 compatible). ADLINK also provides Kernel 2.4.20 (RedHat 8.0 and 9.0 compatible) and Kernel 2.4.22 (RedHat compatible) driver re-compiler service for customers. If users require this service, please contact ADLINK's FAE for more details. For driver support of other Linux releases, please contact ADLINK’s FAE for current support status.

5 Extension Modules

5.1 Extension Modules Overview

Product series

The GEME system is designed to be extendable by one PMC and up to three PC104 modules. This chapter provides information on compatible extension modules:

Function	Bus	Model Name	Description
Motion	PC104	MPC-8164	4-axis stepping and servo motion control card
	PC104	MPC-8372 / 66	12-axis / 6-axis SSCNET servo motion control card
Vision	PMC	PMC-RTV21G	4-CH video capture board for NTSC/PAL cameras
Comm.	PMC	PMC-3534G	4-port RS-232 serial communication module
	PMC	PMC-3544G	4-port RS-422/RS-485 serial communication module
	PMC	PMC-7841G	CAN bus communication card
HSL	PMC	PMC-7852G	High Speed Link master controller interface module
DIO	PC104	MPC-7632	32-CH Digital I/O module
	PC104	MPC-7664	64-CH Digital I/O module

Table 5-1: Product Series

Please consult the relevant manuals for further information on the cards above.

GEME Extension Example

The following figure is an example of a GEME system with extension modules. It shows a GEME system with three PC-104 modules and one PMC module.

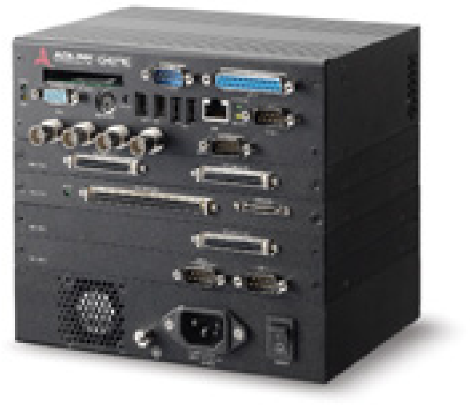


Figure 5-1: GEME w/one PMC and three PC-104 modules

5.2 MPC-8366/8372

Features

MPC-8366/72 are 6/12-axis serial connection motion controllers. They provide the advanced features as follows.

- PC/104 interface
- Servo interface: SSCNET II protocol (Update rate: 0.888 ms)
- Up to 6/12 axes
- 32-bit command resolution
- On-line servo tuning and data monitoring
- Easy wiring up to 30 meters
- Multiple axes linear interpolation
- Any 2 axes circular interpolation
- Contour following motion
- On-the-fly motion/ velocity change
- Programmable interrupt source
- Two 16-bit analog input channels.
- 32-bit external encoder channels
- Two differential pulse output channels
- Software support Windows 2K/XP
- MotionCreatorTM and Trajectory Generator utility

Specifications

The following lists summarize the main specifications of the MPC-8366/72 board motion control system.

	Item	Description
System	Bus Type for PCI board	PCI Rev. 2.2, 33MHz
	Bus Type for MPC board	PC/104
	Bus width for PCI/MPC	32-bit / 16-bit
	Bus Voltage	5V
	Memory usage	16KByte
	IRQ on PCI board	Assigned by PCI controller
	IRQ on MPC board	Assigned by Software Utility
General Specifications	Operating temperature	0°C ~ 60°C
	Storage temperature	-20°C ~ 80°C
	Humidity	5 ~ 95%, non-condensing
	Power Consumption	PCI (MPC)-8372/8366: +5V @ 1 A typical
DSP	Type	TI TMS320C6711
	Clock	100MHz
	DSP performance	600 MFLOPS
Board Interface	I/O Connector	68-pin VHDIC
	SSCNet Connector	3M 10220-52A2JL
Driver Communication	Protocol	SSCNET II
	Bit Rate	5.625Mhz
	Physical layer	RS-485
	Maximum working length	30m for each 6 axes
	Error detection	CRC
Servo Loop	Max. No of controllable axes	8372: 12, 8366: 6
	Servo update rate	0.888ms
	Servo Data Monitors	Current position
		Droop (deviation)
		Velocity Command
		Velocity feedback
		Torque command
		Servo alarm number …etc
	Servo parameter tuning	Parameter read/write

Table 5-2: MPC-8366/72 Specifications

	Item	Description
Motion Function	Motion Velocity Profile	Trapezoidal & S-Curve
	Single motion	Jog move
		Single axis P to P motion
		Change P/V on the fly
		Linear interpolation: up to 4 axes
		2-axis Circular interpolation
	Home move	2 home modes
	Continuous motion	Start / End motion list
		Add linear trajectory
		Add arc trajectory: 2 axes
		Add Dwell
		Smooth Trajectory
		Start/Sop command
		Load Trajectory from file
		Motion I/O status read/configure
		Motion status
Application Functions	Move Ratio	In unit of Pulse per mm
	Software Limit	Each axis has 2 soft limits
	Position Compare	Each axis has 2 comparators
	Interlock	2 axes interlock system
	System error check	Watchdog timer
Interrupt	During operation stop	Possible to select conditions where interrupt occurs
	During alarms, etc.	Yes
Optical Isolated Digital Input	+Limit Switch x 12 (PEL)	Sink or source type are selectable in all channels (all channels must be the same) Input voltage range: 0 ~ 24V Logic H: 14.4~24V Logic L: 0~5V Input resistor: 4.7KW @ 0.5W DI change of state detection Isolated voltage: 500 Vrms Bandwidth: 10K Hz (0.1 ms)
	-Limit Switch x 12 (MEL)	
	Proximity dog x 12 (ORG)	
	General Purposed Input x 2 (PCI board only)	
	Emergency Stop x 1	
Digital Output	DO x 2	Output type: Open-collector (PC3H7)
		Sink Current: 6.5mA Min.
		Isolated voltage: 500 VDC
		Bandwidth: 10K Hz(0.1 ms)

Table 5-2: MPC-8366/72 Specifications

	Item	Description
Analog Out	DA x 2	Resolution: 16 bits
		Settling Time: 10mS Max.
		Output Range: ±10V
		Output Coupling: DC
		Output Impedance: 30W Max.
		Output Driving: ±5mA max.
		Power On State: Floating
		Calibration: Self-Calibration
		Gain Error: ±3% Max.
		Offset Error: 1mV Max. for PCI board 0.2mV Max. for MPC board
Analog In	AD x 2 (Available for MPC/ cPCI board)	Resolution: 16 bits, no missing code
		Sampling Rate: 250kS/s
		Programmable Input Range: ±10V, ±5V, ±2.5V
		Calibration: Self-Calibration
		Gain Error: ±0.03% Max.
		Offset Error: 0.2mV Max.
Encoder Interface	32-bit Encoder input (A,B,Z) x 3 channel (PCI board) 2 channel (MPC/cPCI board)	Incremental Encoder Input
		Max. Speed : 5Mhz
		Input Voltage: 0 - 5V dc
		Logic H: 3-5V
		Logic L: 0-2.4V
		Input resistor: 220O @ 0.125W
		Isolated voltage:500 Vrms
Pulse Output	2 channel differential pulses output (Available for MPC/ cPCI board)	OUT/DIR, CW/CCW, AB phase selectable
		Max. Output Frequency: 4.16 MHz
		Isolated voltage:500 Vrms
Aux. DIO	Board to board synchronous interface (PCI board only)	CN4
	6 TTL Level Digital Output (at CN3 on Extension bracket of PCI board only)	Voltage output high:
		Typical: 5V
		Min: 2.4v @ 15mA
		Voltage output low:
		Typical: 0.3V @ 24mA
		Max: 0.5V

Table 5-2: MPC-8366/72 Specifications

	Item	Description
Extension Bracket	Optional bracket for SSCNET Axis 7-12 splitter and TTL Level Digital Output for PCI-8372 only	CN3 (A) , CN2 (A)

Table 5-2: MPC-8366/72 Specifications

SP1 Pin Assignment: MPC-8372/66 I/O Connector

No.	Name	I/O	Function Axis	No.	Name	I/O	Function Axis
1	DO_COM	-	Common for Digital Output	35	DO1	O	General Digital Output
2	PEL1/MDI1	I	Positive End Limit	36	DO2	O	General Digital Output
3	MEL1/MDI2	I	Minus End Limit	37	PEL2/MDI4	I	Positive End Limit
4	ORG1/MDI3	I	Origin Signal	38	MEL2/MDI5	I	Minus End Limit
5	PEL3/MDI7	I	Positive End Limit	39	ORG2/MDI6	I	Origin Signal
6	MEL3/MDI8	I	Minus End Limit	40	PEL4/MDI10	I	Positive End Limit
7	ORG3/MDI9	I	Origin Signal	41	MEL4/MDI11	I	Minus End Limit
8	PEL5/MDI13	I	Positive End Limit	42	ORG4/MDI12	I	Origin Signal
9	MEL5/MDI14	I	Minus End Limit	43	PEL6/MDI16	I	Positive End Limit
10	ORG5/MDI15	I	Origin Signal	44	MEL6/MDI17	I	Minus End Limit
11	IPT_COM/EMG_COM	-	Common for Digital Input	45	ORG6/MDI18	I	Origin Signal
12	EA1+	I	Encoder A-Phase (+)	46	EA2+	I	Encoder A-Phase (+)
13	EA1-	I	Encoder A-Phase (-)	47	EA2-	I	Encoder A-Phase (-)
14	EB1+	I	Encoder B-Phase (+)	48	EB2+	I	Encoder B-Phase (+)
15	EB1-	I	Encoder B-Phase (-)	49	EB2-	I	Encoder B-Phase (-)
16	EZ1+	I	Encoder Z-Phase (+)	50	EZ2+	I	Encoder Z-Phase (+)
17	EZ1-	I	Encoder Z-Phase (-)	51	EZ2-	I	Encoder Z-Phase (-)
18	PEL7/MDI19	I	Positive End Limit	52	PEL8/MDI22	I	Positive End Limit
19	MEL7/MDI20	I	Minus End Limit	53	MEL8/MDI23	I	Minus End Limit
20	ORG7/MDI21	I	Origin Signal	54	ORG8/MDI24	I	Origin Signal
21	PEL9/MDI25	I	Positive End Limit	55	PEL10/MDI28	I	Positive End Limit
22	MEL9/MDI26	I	Minus End Limit	56	MEL10/MDI29	I	Minus End Limit
23	ORG9/MDI27	I	Origin Signal	57	ORG10/MDI30	I	Origin Signal
24	PEL11/MDI31	I	Positive End Limit	58	PEL12/MDI34	I	Positive End Limit
25	MEL11/MDI32	I	Minus End Limit	59	MEL12/MDI35	I	Minus End Limit
26	ORG11/MDI33	I	Origin Signal	60	ORG12/MDI36	I	Origin Signal
27	IPT_COM/EMG_COM	-	Common for Digital Input	61	EMG	I	Emergency Stop Signal
28	P_GND	-	Common for Pulse Interface	62	AD1	I	Analog Input
29	OUT1+	O	Pulse signal (+)	63	DIR1+	O	Dir. signal (+)
30	OUT1-	O	Pulse signal (-)	64	AD2	I	Analog Input
31	OUT2+	O	Pulse signal (+)	65	DIR1-	O	Dir. signal (-)
32	OUT2-	O	Pulse signal (-)	66	DA1	O	Analog Output
33	DIR2+	O	Dir. signal (+)	67	DA2	O	Analog Output
34	DIR2-	O	Dir. signal (-)	68	A_COM	-	Analog Ground

Table 5-3: SP1 Pin Assignment: MPC-8372/66 I/O Connector

MDI# is for general purpose input if it is not used for motion.

Dimensions

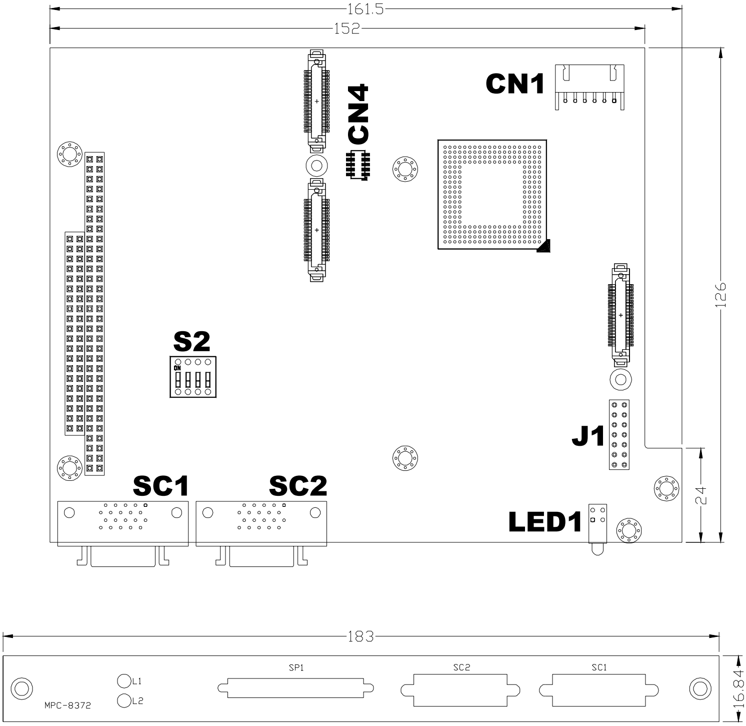


Figure 5-2: MPC-8372 PCB Layout and Front Panel

SC1: SSCNET connector for Axis 0~5

SC2: SSCNET connector for Axis 6~11

SP1: Daughter Board connector

LED1: Board Status LEDs

S2: DIP switch for I/O address setting

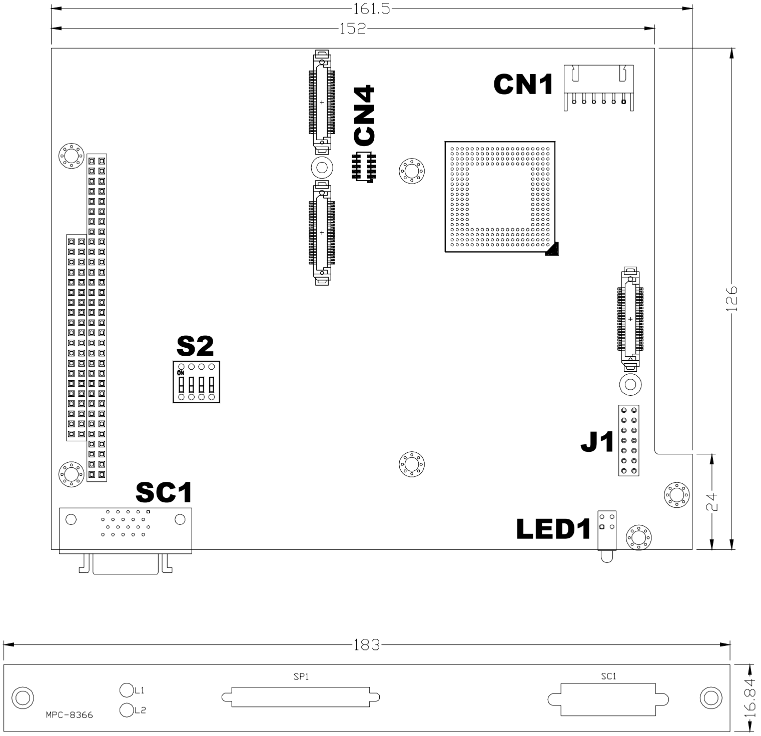


Figure 5-3: MPC-8366 PCB Layout and Front Panel

SC1: SSCNET connector for Axis 0~5

SP1: Daughter Board connector

LED1: Board Status LEDs

S2: DIP switch for I/O address setting

5.3 MPC-8164

Features

- 16-bit PC104 Bus
- Axes of step and direction pulse output for controlling stepping or servomotor
- Maximum output frequency of 6.55 MPPS
- Pulse output options: OUT/DIR, CW/CCW
- Programmable acceleration and deceleration time for all modes
- Trapezoidal and S-curve velocity profiles for all modes
- Any 2 of 4 axes circular interpolation
- Any 2-4 of 4 axes linear interpolation
- Continuous interpolation for contour following motion
- Change position and speed on the fly
- Change speed by comparator condition
- 13 home return modes with auto searching
- Hardware backlash compensator and vibration suppression
- Software end-limits for each axis
- 28-bit up/down counter for incremental encoder feedback
- Home switch, index signal(EZ), positive, and negative end limit switches interface on all axes
- 2-axis high speed position latch input
- 2-axis position compare trigger output with 4k FIFO auto-loading
- All digital input and output signals are 2500Vrms isolated
- Programmable interrupt sources
- Eight channels of general purpose photo-isolated digital inputs
- Eight channels of general purpose open collector digital outputs
- Software supports a maximum of up to four MPC-8164 cards (16 axes) operation in one system
- Includes Motion Creator, Microsoft Windows-based application development software
- MPC-8164 Libraries and Utilities for DOS and Windows 98/ NT/2000/XP. Also supports Windows XP/NT Embedded
- MPC-8164 Libraries for Linux and Windows CE systems

CN2 Pin Assignments: Main connector

CN2 is the major connector for the motion control I/O signals.

No.	Name	I/O	Function (axis (1) / (2))	No.	Name	I/O	Function (axis(3) / (4))
1	VPP	O	+5V power supply output	51	VPP	O	+5V power supply output
2	GND		Ext. power ground	52	GND		Ext. power ground
3	OUT1+	O	Pulse signal (+), (1)	53	OUT3+	O	Pulse signal (+), (3)
4	OUT1-	O	Pulse signal (-), (1)	54	OUT3-	O	Pulse signal (-), (3)
5	DIR1+	O	Dir. signal (+), (1)	55	DIR3+	O	Dir. signal (+), (3)
6	DIR1-	O	Dir. signal (-), (1)	56	DIR3-	O	Dir. signal (-), (3)
7	SVON1	O	Multi-purpose signal, (1)	57	SVON3	O	Multi-purpose signal, (3)
8	ERC1	O	Dev. ctr, clr. signal, (1)	58	ERC3	O	Dev. ctr, clr. signal, (3)
9	ALM1	I	Alarm signal, (1)	59	ALM3	I	Alarm signal, (3)
10	INP1	I	In-position signal, (1)	60	INP3	I	In-position signal, (3)
11	RDY1	I	Multi-purpose signal, (1)	61	RDY3	I	Multi-purpose signal, (3)
12	GND		Ext. power ground	62	EXGND		Ext. power ground
13	EA1+	I	Encoder A-phase (+), (1)	63	EA3+	I	Encoder A-phase (+), (3)
14	EA1-	I	Encoder A-phase (-), (1)	64	EA3-	I	Encoder A-phase (-), (3)
15	EB1+	I	Encoder B-phase (+), (1)	65	EB3+	I	Encoder B-phase (+), (3)
16	EB1-	I	Encoder B-phase (-), (1)	66	EB3-	I	Encoder B-phase (-), (3)
17	EZ1+	I	Encoder Z-phase (+), (1)	67	EZ3+	I	Encoder Z-phase (+), (3)
18	EZ1-	I	Encoder Z-phase (-), (1)	68	EZ3-	I	Encoder Z-phase (-), (3)
19	VPP	O	+5V power supply output	69	VPP	O	+5V power supply output
20	GND		Ext. power ground	70	GND		Ext. power ground
21	OUT2+	O	Pulse signal (+), (2)	71	OUT4+	O	Pulse signal (+), (4)
22	OUT2-	O	Pulse signal (-), (2)	72	OUT4-	O	Pulse signal (-), (4)
23	DIR2+	O	Dir. signal (+), (2)	73	DIR4+	O	Dir. signal (+), (4)
24	DIR2-	O	Dir. signal (-), (2)	74	DIR4-	O	Dir. signal (-), (4)
25	SVON2	O	Multi-purpose signal, (2)	75	SVON4	O	Multi-purpose signal, (4)
26	ERC2	O	Dev. ctr, clr. signal, (2)	76	ERC4	O	Dev. ctr, clr. signal, (4)
27	ALM2	I	Alarm signal, (2)	77	ALM4	I	Alarm signal, (4)
28	INP2	I	In-position signal, (2)	78	INP4	I	In-position signal, (4)
29	RDY2	I	Multi-purpose signal, (2)	79	RDY4	I	Multi-purpose signal, (4)
30	GND		Ext. power ground	80	GND		Ext. power ground
31	EA2+	I	Encoder A-phase (+), (2)	81	EA4+	I	Encoder A-phase (+), (4)
32	EA2-	I	Encoder A-phase (-), (2)	82	EA4-	I	Encoder A-phase (-), (4)

Table 5-4: CN2 Pin Assignments: Main connector

No.	Name	I/O	Function (axis (1) / (2))	No.	Name	I/O	Function (axis(3) / (4))
33	EB2+	I	Encoder B-phase (+), (2)	83	EB4+	I	Encoder B-phase (+), (4)
34	EB2-	I	Encoder B-phase (-), (2)	84	EB4-	I	Encoder B-phase (-), (4)
35	EZ2+	I	Encoder Z-phase (+), (2)	85	EZ4+	I	Encoder Z-phase (+), (4)
36	EZ2-	I	Encoder Z-phase (-), (2)	86	EZ4-	I	Encoder Z-phase (-), (4)
37	PEL1	I	End limit signal (+), (1)	87	PEL3	I	End limit signal (+), (3)
38	MEL1	I	End limit signal (-), (1)	88	MEL3	I	End limit signal (-), (3)
39	CMP1	O	Position compare output (1)	89	LTC3	I	Position latch input (3)
40	SD/PCS1	I	Ramp-down signal (1)	90	SD/PCS3	I	Ramp-down signal (3)
41	ORG1	I	Origin signal, (1)	91	ORG3	I	Origin signal, (3)
42	GND		Ext. power ground	92	GND		Ext. power ground
43	PEL2	I	End limit signal (+), (2)	93	PEL4	I	End limit signal (+), (4)
44	MEL2	I	End limit signal (-), (2)	94	MEL4	I	End limit signal (-), (4)
45	CMP2	O	Position compare output (2)	95	LTC4	I	Position latch input, (4)
46	SD/PCS2	I	Ramp-down signal (2)	96	SD/PCS4	I	Ramp-down signal (4)
47	ORG2	I	Origin signal, (2)	97	ORG4	I	Origin signal, (4)
48	GND		Ext. power ground	98	GND		Ext. power ground
49	GND		Ext. power ground	99	E_24V		Ext. power supply, +24V
50	GND		Ext. power ground	100	E_24V		Ext. power supply, +24V

Table 5-4: CN2 Pin Assignments: Main connector

CN3 Pin Assignment: General Purpose DI/DO ports

CN3 Pin	Signal Name	CN3 Pin	Signal Name
1	DOCOM	2	DOCOM
3	DOCOM	4	DOCOM
5	DO0	6	DO1
7	DO2	8	DO3
9	DO4	10	DO5
11	DO6	12	DO7
13	--	14	DICOM
15	DICOM	16	DICOM
17	DICOM	18	DI0
19	DI1	20	DI2
21	DI3	22	DI4
23	DI5	24	DI6
25	DI7	26	--

Table 5-5: CN3 Pin Assignment - General Purpose DI/DO

MPC-8164 Dimensions

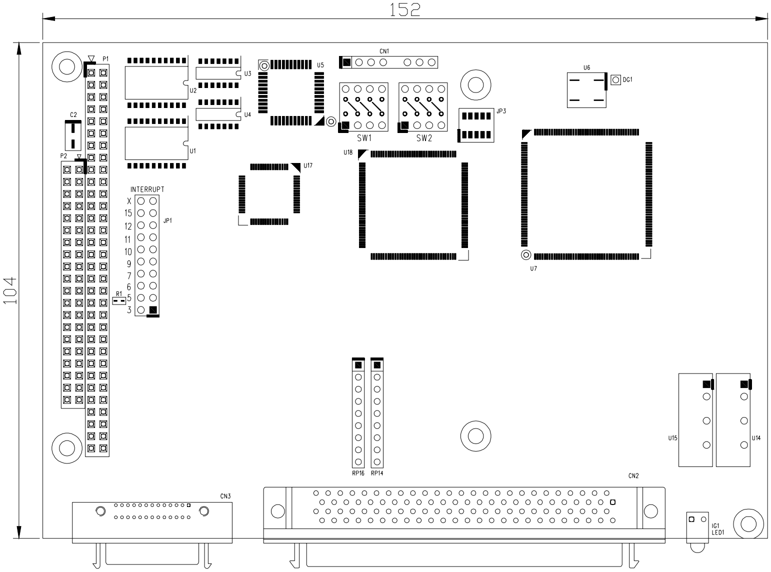


Figure 5-4: MPC-8164 PCB Layout

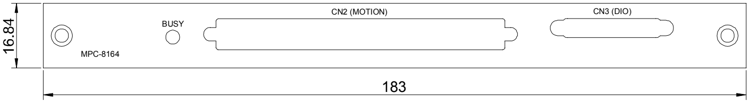


Figure 5-5: MPC-8164 Front Panel

5.4 MPC-7632/7632AU/7664

Features

MPC-7632/7632AU/7664 isolated DIO cards provide the following advanced features:

- PC/104 interface
- 16/32 channels isolated digital input channel
- 16/32 channels isolated digital output channel
- High output current (80mA per channel)
- 2500 VRMS voltage isolation
- One external interrupt channel
- Inputs with change-of-state function
- High-level language function libraries
- Software supports DOS, Windows 98/NT/2K/XP, Linux 2.4 or higher, Windows CE, and Windows XP Embedded.

	Parameter	Value
Optical Isolated Input Channels	Number of channels*	16 (MPC-7632) 32 (MPC-7664)
	External interrupt channels	1
	Input voltage	DC12V to 24V (±10%)
	Input current	5 to 15mA/bit (Max)
	Turn-on time (off → on)	3.5us (Typ)
	Turn-off time (on → off)	50us (Typ)

Table 5-6: MPC-7632/7632AU/7664 Features

	Parameter	Value
Optical Isolated Output Channels	Number of channels	16 (MPC-7632) 32 (MPC-7664)
	Voltage between terminals	DC30V (Max)
	Output current	80mA(Max)
	Output voltage drop	1V(Max)
	Turn-on time (off → on)	2.8us (Typ)
	Turn-off time (on → off)	400us (Typ)
General Specifications	Current consumption	400mA @ +5V (±5%)
	Isolation voltage	2.5kVRMS (Min.)
Environment Condition	Operating Temperature	0 to 50°C
	Operating Humidity	35 to 85%
Audio**	THD+N @ 1KHz	0.1%(Max) @ 200mW into 8Ω 0.1%(Typ) @ 85mW into 32Ω

Table 5-6: MPC-7632/7632AU/7664 Features

* The first three input channels provide the Change-of-State functionality

** Audio spec for MPC-7632AU only

CP1 Pin Assignment

The pin assignment of the 50-pin SCSI connector CP1 for the 7632/7632AU/7664 is shown below.

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	INTCOM	-	Common for interrupt source	26	INT	I	Interrupt source input
2	DI0	I	Digital input	27	DICOM0	-	Common for digital input
3	DI1	I	Digital input	28	DI2	I	Digital input
4	DICOM0	-	Common for digital input	29	DI3	I	Digital input
5	DI4	I	Digital input	30	DICOM0	-	Common for digital input
6	DI5	I	Digital input	31	DI6	I	Digital input
7	DICOM0	-	Common for digital input	32	DI7	I	Digital input
8	DI8	I	Digital input	33	DICOM1	-	Common for digital input
9	DI9	I	Digital input	34	DI10	I	Digital input
10	DICOM1	-	Common for digital input	35	DI11	I	Digital input
11	DI12	I	Digital input	36	DICOM1	-	Common for digital input
12	DI13	I	Digital input	37	DI14	I	Digital input
13	DICOM1	-	Common for digital input	38	DI15	I	Digital input
14	DO0	O	Digital output	39	DOCOM0	-	Common for digital output
15	DO1	O	Digital output	40	DO2	O	Digital output
16	DOCOM0	-	Common for digital output	41	DO3	O	Digital output
17	DO4	O	Digital output	42	DOCOM0	-	Common for digital output
18	DO5	O	Digital output	43	DO6	O	Digital output
19	DOCOM0	-	Common for digital output	44	DO7	O	Digital output
20	DO8	O	Digital output	45	DOCOM1	-	Common for digital output
21	DO9	O	Digital output	46	DO10	O	Digital output
22	DOCOM1	-	Common for digital output	47	DO11	O	Digital output
23	DO12	O	Digital output	48	DOCOM1	-	Common for digital output
24	DO13	O	Digital output	49	DO14	O	Digital output
25	DOCOM1	-	Common for digital output	50	DO15	O	Digital output

Table 5-7: CP1 Pin Assignment

CP2 Pin Assignment

The pin assignment of the 50-pin SCSI connector CP2 for the 7664 is shown below.

No.	Name	I/O	Function of Axis	No.	Name	I/O	Function of Axis
1	NC	-	No connect	26	NC	I	No connect
2	DI16	I	Digital input	27	DICOM2	-	Common for digital input
3	DI17	I	Digital input	28	DI18	I	Digital input
4	DICOM2	-	Common for digital input	29	DI19	I	Digital input
5	DI20	I	Digital input	30	DICOM2	-	Common for digital input
6	DI21	I	Digital input	31	DI22	I	Digital input
7	DICOM3	-	Common for digital input	32	DI23	I	Digital input
8	DI24	I	Digital input	33	DICOM3	-	Common for digital input
9	DI25	I	Digital input	34	DI26	I	Digital input
10	DICOM3	-	Common for digital input	35	DI27	I	Digital input
11	DI28	I	Digital input	36	DICOM3	-	Common for digital input
12	DI29	I	Digital input	37	DI30	I	Digital input
13	DICOM3	-	Common for digital input	38	DI31	I	Digital input
14	DO16	O	Digital output	39	DOCOM2	-	Common for digital output
15	DO17	O	Digital output	40	DO18	O	Digital output
16	DOCOM2	-	Common for digital output	41	DO19	O	Digital output
17	DO20	O	Digital output	42	DOCOM2	-	Common for digital output
18	DO21	O	Digital output	43	DO22	O	Digital output
19	DOCOM2	-	Common for digital output	44	DO23	O	Digital output
20	DO24	O	Digital output	45	DOCOM3	-	Common for digital output
21	DO25	O	Digital output	46	DO26	O	Digital output
22	DOCOM3	-	Common for digital output	47	DO27	O	Digital output
23	DO28	O	Digital output	48	DOCOM3	-	Common for digital output
24	DO29	O	Digital output	49	DO30	O	Digital output
25	DOCOM3	-	Common for digital output	50	DO31	O	Digital output

Table 5-8: CP2 Pin Assignment

Dimensions

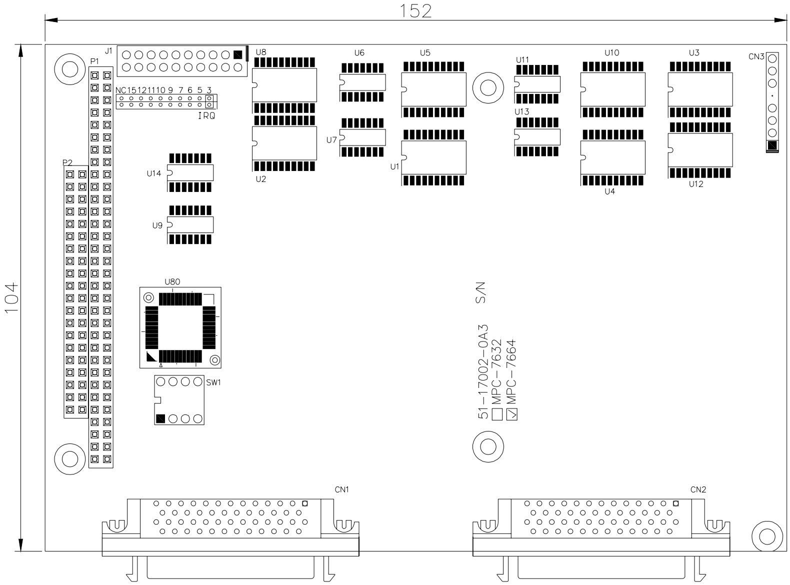


Figure 5-6: MPC-7664 PCB layout

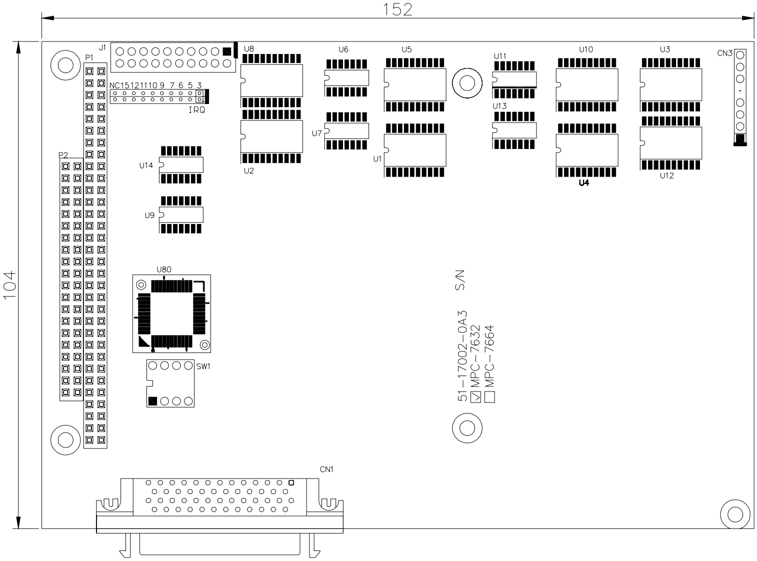


Figure 5-7: MPC-7632 PCB layout

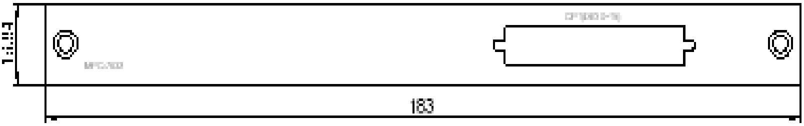


Figure 5-8: MPC-7632 front panel

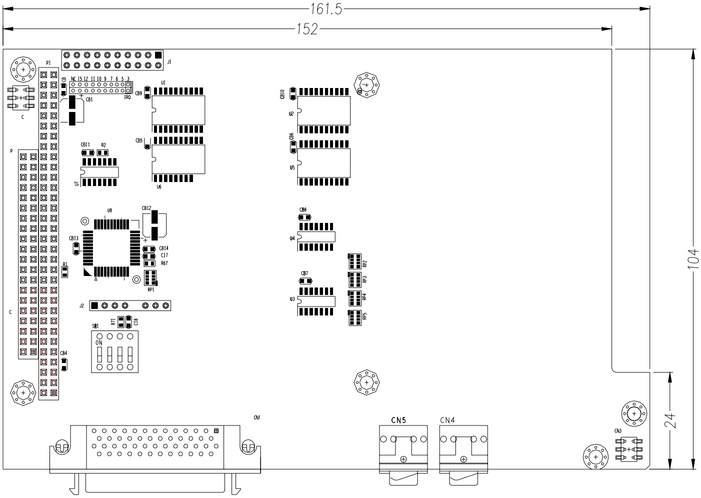


Figure 5-9: MPC-7632AU PCB layout

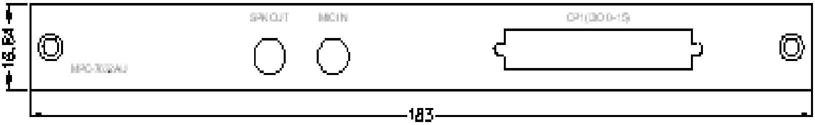


Figure 5-10: MPC-7632AU front panel

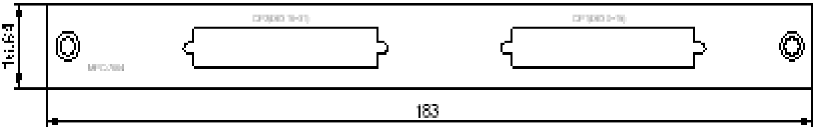


Figure 5-11: MPC-7664 front panel

5.5 PMC-RTV24G

Features

- 4-channel full-frame acquisition from single video stream
- Up to 30fps in 32-bit, 33MHz PMC bus
- Color (PAL / NTSC), monochrome (CCIR / EIA) cameras
- On-board TTL I/O lines
- User-friendly ViewCreator utility
- Software trigger supported

Specifications

Video Input

- Four composite video color digitizers
- Video input interface: 10-pin header connectors
- Coaxial cable recommended

General Purpose I/O Lines

The I/O lines are TTL compatible with 1 input and 1 output

GPIO interface:

- One 10-pin header connector
- I/O lines are internally pulled up and have the following characteristics:

Voltage	MIN	MAX
Input high voltage (20μA)	2.0V	5.25V
Input low voltage (-0.2mA)	0.0V	0.80V
Output high voltage (-1.0mA)	5.0V	-
Output low voltage (100.0mA)	-	0.5V

Table 5-9: GPIO Interface Voltage

User EEPROM

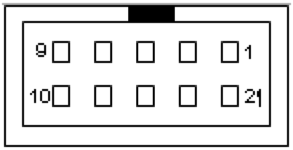
Includes 1kbit available EEPROM

Form Factor

32-bit / 33MHz PMC socket board

Pin Assignment

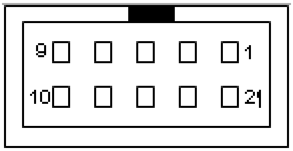
Video Input



PIN NO.	Function	PIN NO.	Function
1	GND	2	CH0 video in
3	CH1video in	4	GND
5	GND	6	CH2video in
7	CH3video in	8	GND
9	GND	10	GND

Table 5-10: Video Input Connector

GPIO



PIN NO.	Function	PIN NO.	Function
1	IN0 External interrupt	2	GND
3	OUT0	4	NC
5	NC	6	GND
7	NC	8	+5V
9	GND	10	NC

Table 5-11: GPIO Connector

Dimensions

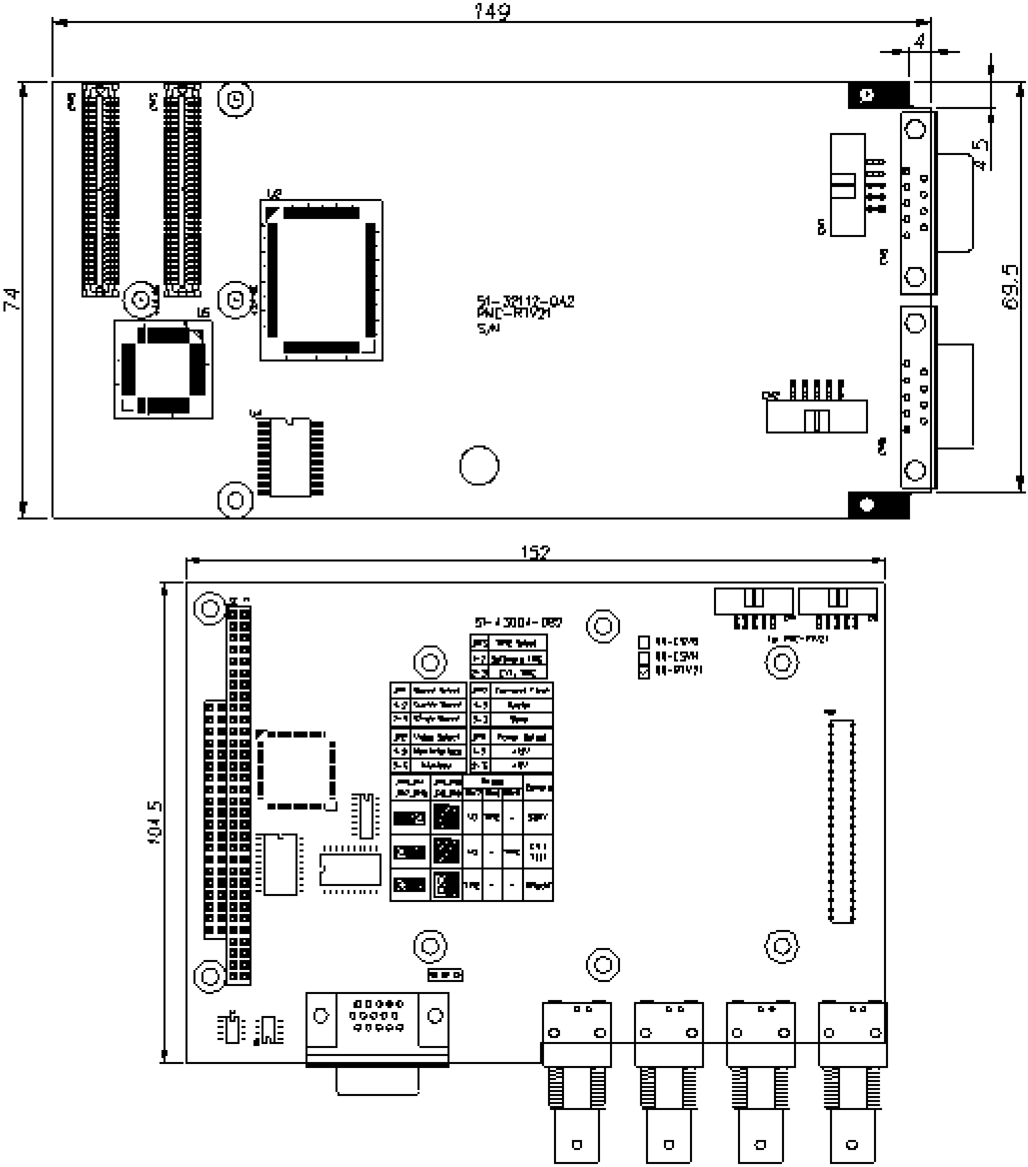


Figure 5-12: PCB layout of the PMC-RTV24G & DB-RTV24G

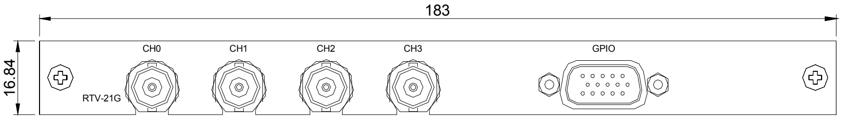


Figure 5-13: Front panel of PMC-RTV24G for GEME system

5.6 PMC-3534G

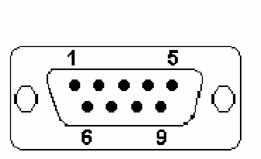
Features

- IRQ and I/O address automatically assigned by PCI Plug and Play
- Four RS-232C communication ports with intelligent buffer
- High speed concurrent communication (max. 115200bps)
- Suitable for modems, data display, data collection, telecommunication

Specifications

- Compliant with PCI Spec.2.1
- Serial communication controller:
  - 16C550A compatible
- System I/O mapping:
  - Assigned by PCI BIOS
- Shared IRQ
- Flow control
- Xon/Xoff control
- RTS/CTS control
- Port Capability: independent RS-232C compatible ports
- Baud rate: Each port can be configured to 50-115,200 bps

Pin Assignment



Pin	RS-232
1	DCD, Data carrier detect
2	RXD, Receive data
3	TXD, Transmit data
4	DTR, Data terminal ready
5	GND, ground
6	DSR, Data set ready
7	RTS, Request to send
8	CTS, Clear to send
9	RI, Ring indicator

Table 5-12: PMC-3534G Pin Assignment

Dimensions

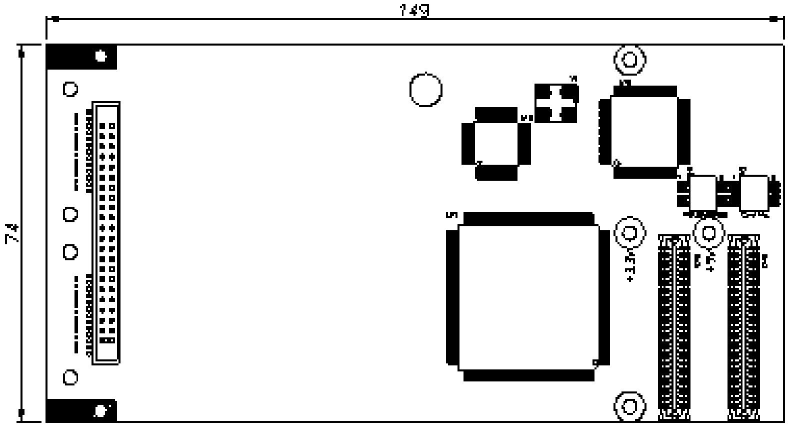


Figure 5-14: PMC-3534G Dimensions

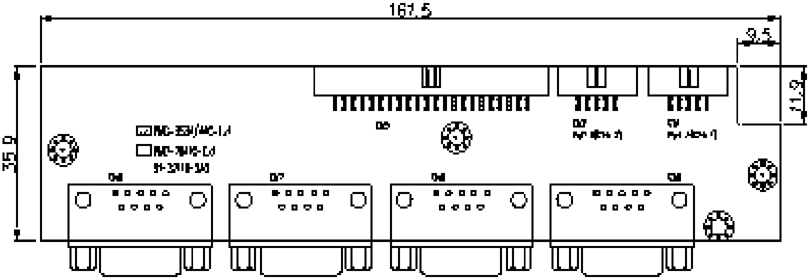


Figure 5-15: PMC-3534G PCB layout and extension card

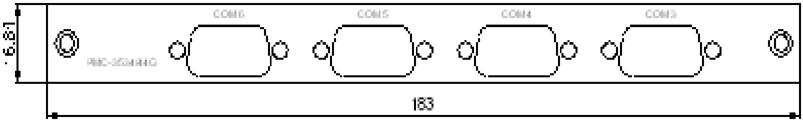


Figure 5-16: PMC-3534G Front panel

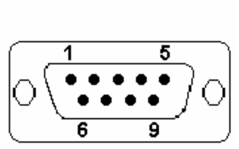
5.7 PMC-3544G

Features

- IRQ and I/O address automatically assigned by PCI Plug and Play
- Communication ports with intelligent buffer
- RS-422/485 hardware selectable
- RS-485 with auto direction flow control
- High speed concurrent communications (max. 115200bps)

Specifications

- Compliant with PCI Spec.2.1
- Serial communication controller:
- 16C550A compatible
- System I/O mapping: assigned by PCI BIOS
- Shared IRQ
- Flow control
- RS-485 auto direction
- Port Capability:
- Four channel RS-422/485 port (DIP switch select-for DIP switch configuration, please refer to the EBC board manual)
- Baud rate: Each port can be configured to 50-115,200 bps



Pin	RS-422	RS-485
1	RX+	NC
2	TX+	NC
3	NC	NC
4	NC	DATA+
5	GND	GND
6	RX-	NC
7	TX-	NC
8	NC	NC
9	NC	DATA-

Table 5-13: PMC-3544G

Dimensions

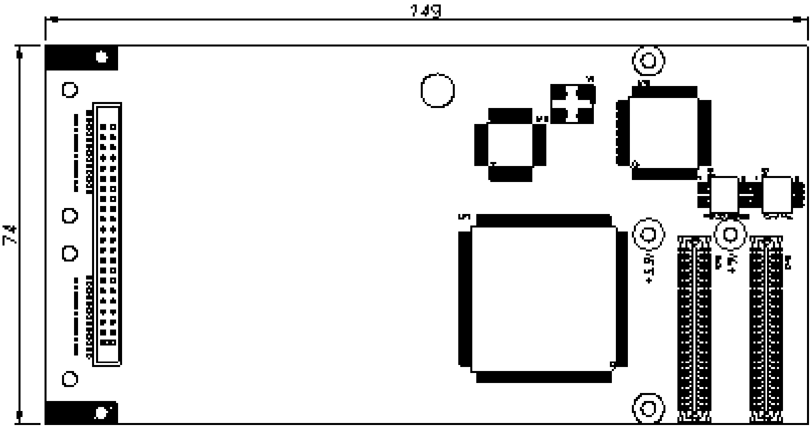


Figure 5-17: PMC-3544G Dimensions

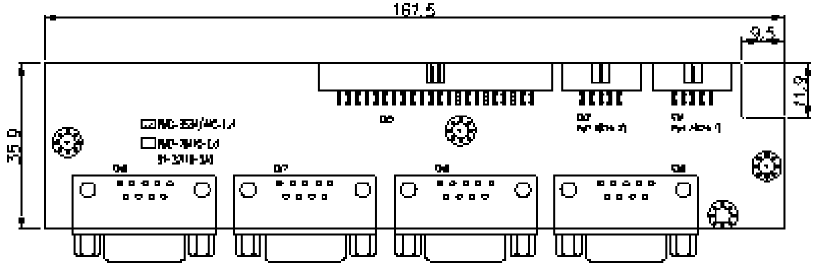


Figure 5-18: PMC-3544G PCB layout and extension card

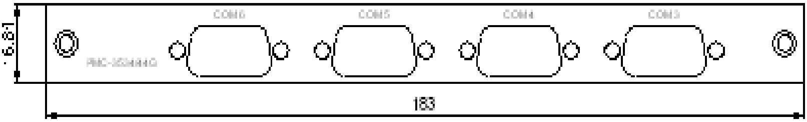


Figure 5-19: PMC-3544G front panel

5.8 PMC-7841G

Features

The PMC-7841G is a Dual-Port Isolated CAN Interface Card with the following features:

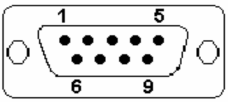
- Two independent CAN network operation
- Bridge support
- Compatible with CAN specification 2.0 parts A and B
- Optically isolated CAN interface (up to 2500 Vrms isolation protection)
- Direct memory mapping to the CAN controllers
- Up to 1Mbps programmable transfer rate

Specifications

Ports	2 CAN channels (V2.0 A, B)
CAN Controller	SJA1000
CAN Transceiver	82c250
Signal Support	CAN_H, CAN_L
Isolation Voltage	2500 Vrms
Operation Temperature	0 to 60ºC
Storage Temperature	-20 to 80ºC
Humidity	5 to 95% non-condensing
IRQ Level	Set by Plug and Play BIOS
I/O port address	Set by Plug and Play BIOS
Power Consumption (without external devices)	400mA @ 5VDC (Typical) 900mA @ 5VDC (Maximum)

Table 5-14: PMC-7841G Specifications

Pin Assignment



Pin	CAN
1	NC
2	CAN_L
3	Shield
4	NC
5	Case GND
6	NC
7	CAN_H
8	NC
9	NC

Table 5-15: PMC-7841G Pin Assignment

Dimensions

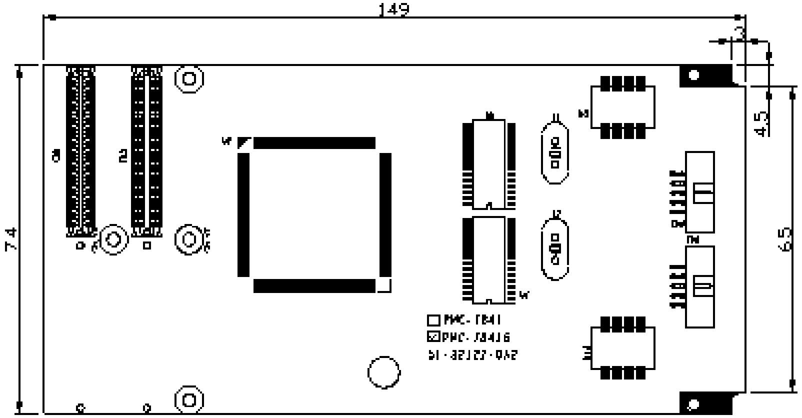


Figure 5-20: PMC-7841G Dimensions

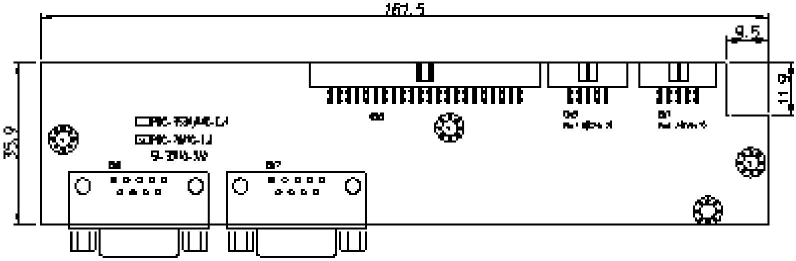


Figure 5-21: PMC-7841G PCB layout and extension card

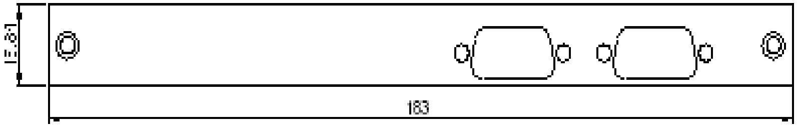


Figure 5-22: PMC-7841G front panel

5.9 PMC-7852G

Features

General

- One master has two HSL ports
- One port can drive a maximum of 32 modules
- One master can control maximum 63 slave I/O modules
- Maximum wiring distance for each port: 200m (serial wiring from master to last slave module)

Wiring:

- Connector: RJ45 (on both master controller and slave modules)
- Cable: Cat.5 100 Base/TX Ethernet cable, shielded preferred

Communications:

- Multi-drop full-duplex RS-422 with transformer isolation scheme
- Data Rate: 6Mbps
- I/O refresh rate: 30.1μs x numbers of slave I/O modules (min: 3; max: 63)
- Communication model: single-master/multi-slave
- Communication method: command-response
- CRC12 and dedicate protocol for eliminating communication errors

Specifications

PCI Controller:

- PCI local bus specification Rev. 2.1 compliance

Master Controller:

- Master controller: ASIC
- External Clock: 48MHz

Memory:

- 32KB SRAM - 12ns

Interface:

- RS-422 with transformer isolation
- Full duplex communication
- Selectable transfer speed by dip switch (Default 6Mbps)
- Two ports for one control master

Connector:

- Four RJ45 connectors (H1A, H1B, H2A, H2B for PMC-7852G)

Interrupt:

- 32 bits Programmable timer

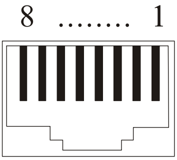
LED Indicator: Power status

Operating Temperature: 0 to 600ºC

Storage Temperature: -20 to 800ºC

Power Consumption: +5V @ 500 mA typical

Pin Assignment



Pin	Pin out
PIN 1	NC
PIN 2	NC
PIN 3	RX+
PIN 4	TX-
PIN 5	TX+
PIN 6	RX-
PIN 7	NC
PIN 8	NC

Table 5-16: PMC-7852G Pin Assignment

Dimensions

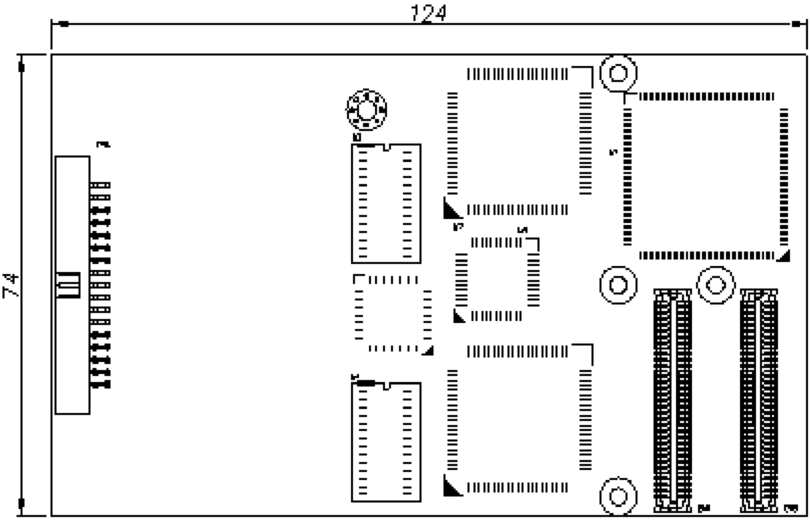


Figure 5-23: PMC-7852G Dimensions

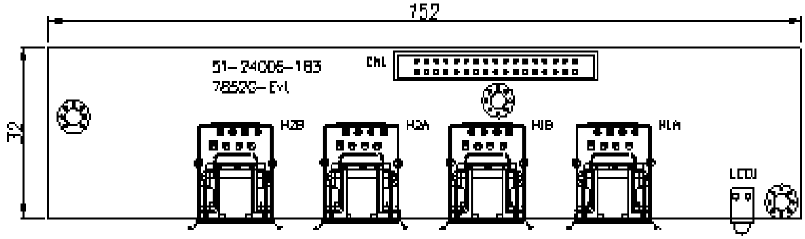


Figure 5-24: PMC-7852G PCB layout and extension module

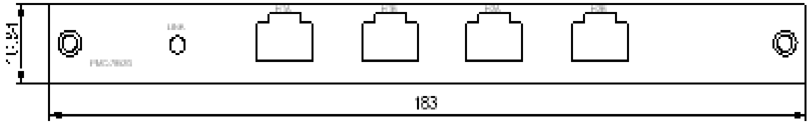


Figure 5-25: PMC-7852G front panel

Appendix: GEME-VM3000 Series Introduction

The GEME-VM3000 series is a new MPEG4 software video compression development platform, which combines GEME-V3000's excellent real-time image acquisition functionality with advanced MPEG4 software video compression algorithms for security and remote video surveillance applications.

The MPEG4 software video compression function library provides high quality video encoding and decoding. Image quality and bitrate are adjustable for more efficient data transmission via TCP/IP. Highly sensitive partial or whole image motion detection for smart video encoding or alarm signaling and 4X image expansion technology for enhanced decoded image quality is provided.

6.1 Features

Image Acquisition

Acquisition speed

NTSC	1 CH	2 CH	3 CH	4 CH
Fields	60	14	10	8
frames/sec/CH	30	7	5	4
PAL	1 CH	2 CH	3 CH	4 CH
Fields	50	12	8	6
frames/sec/CH	25	6	4	3

Color Image: The acquired color video is compatible with the following composite video input formats: NTSC-M, NTSC-Japan, PCL-B, PAL-D, PAL-G, PAL-H, PAL-I, PAM-M, PAL-N, and SECAM

Monochrome Image: The acquired monochrome video is compatible with CCIR and EIA (RS-170)

Optional scaling: Scaling of acquired image or portions of an image is available as follows:

- Adjustment of hue (for NTSC signals), contrast (0 to 200%), brightness, and saturation (0 to 200% for U and V signals).
- Automatic chrominance gain control.

MPEG4 Encoding

MPEG4 video encoding: Video encoding quality level can be set as follows:

Quality	Value	Image in 320*240	Image in 160*120
Lowest	-2	Bitrate = 320000 frame_rate = 3	Bitrate = 240000/4 frame_rate = 3
Low	-1	Bitrate = 400000 frame_rate = 6	Bitrate = 400000/4 frame_rate = 6
Normal	0	Bitrate = 480000 frame_rate = 15	Bitrate = 480000/4 frame_rate = 15
High	1	Bitrate = 512000 frame_rate = 30	Bitrate = 512000/4 frame_rate = 30
Highest	2	Bitrate = 1024000 frame_rate = 30	Bitrate = 1024000/4 frame_rate = 30

Table 6-1: MPEG4 Video Encodings

Supports setting of initial motion detection area and assignment or creation of directory for saved files.

Save video file: A continuous video can be saved to either M4V or AVI video file format. Users may play AVI files with MS Media Player after installing the XVID CODEC (see ADLINK All-in-One CD).

Save single image file: Individual images can be saved to either BMP or JPEG image file format.

MPEG4 Decoding

MPEG4 video decoding: Video decoding can be assigned a source from the local memory buffer, file access, or TCP/IP port. The quality of decoded video can be used to adjust the video encoding level. Decoding quality can also be used to control the flow rate between encoder and decoder and to adjust motion detection settings.

Save video file: A continuous video can be saved to either M4V or AVI video file format. Users may play AVI files with MS Media

Player after installing the XVID CODEC (see ADLINK All-in-One CD).

Save single image file: Individual images can be saved to either BMP or JPEG image file format.

Motion Detection

Up to four detection areas can be set in one frame or the entire frame can be used for motion detection actions. The criteria for motion detection occurrence can be adjusted for sensitivity.

TCP/IP Data Transmission

Video data can be transferred by TCP/IP after connecting to the IP of the encoding site and data transmission speed can monitored using the flow rate.

Supported software

WDM driver: Supports VC++/VB/BCB/Delphi programming under Windows NT/98/2000/XP platforms with DLL.

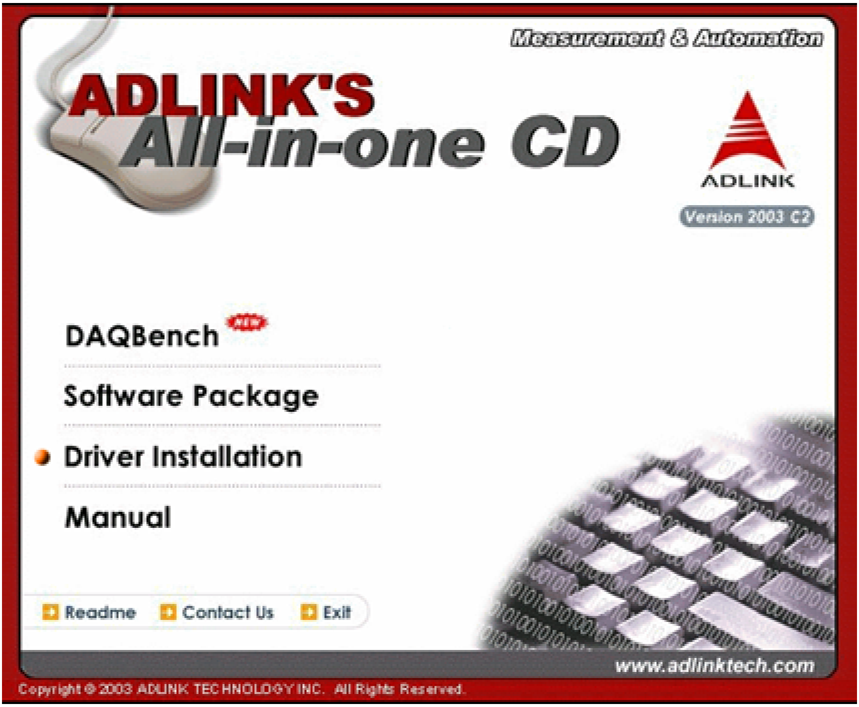
ViewCreator: This package will assist in initial testing and functional evaluation.

6.2 Driver Installation Guide

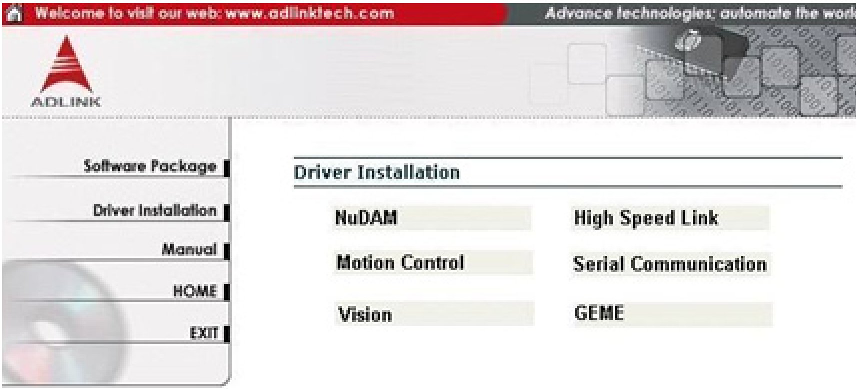
The following guides are for Windows 98/NT/2000/XP/Embedded XP.

Driver Installation for Windows 98/NT/2000/XP

1. Insert the Automation All-in-one CD into the CD-ROM drive and click Driver Installation.



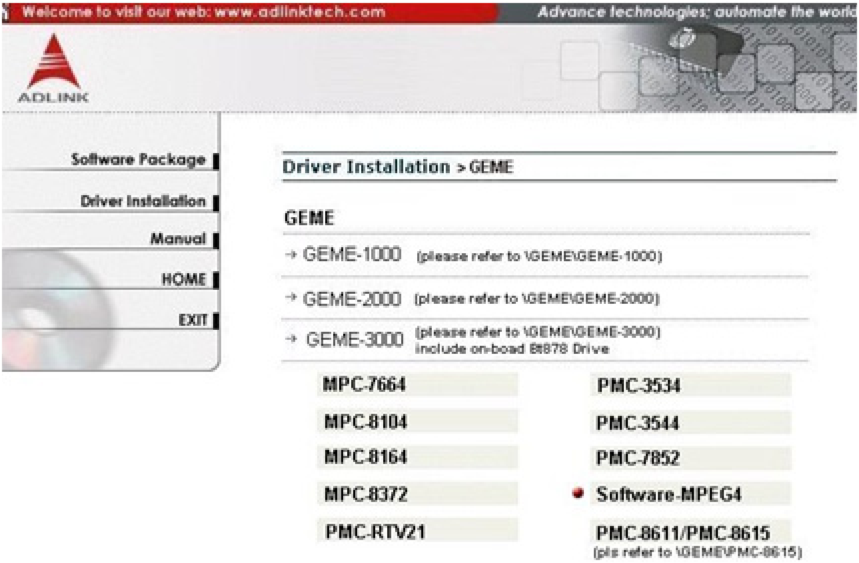
2. Select GEME


Welcome to visit our web: www.adlinktech.com
Advance technologies; automate the world
ADLINK
Software Package
Driver Installation
Manual
HOME
EXIT

Driver Installation

NuDAM | High Speed Link
Motion Control | Serial Communication
Vision | GEME

3. Click Software-MPEG4


Welcome to visit our web: www.adlinktech.com
Advance technologies; automate the world
ADLINK
Software Package
Driver Installation
Manual
HOME
EXIT

Driver Installation > GEME

GEME

→ GEME-1000 (please refer to \GEME\GEME-1000)

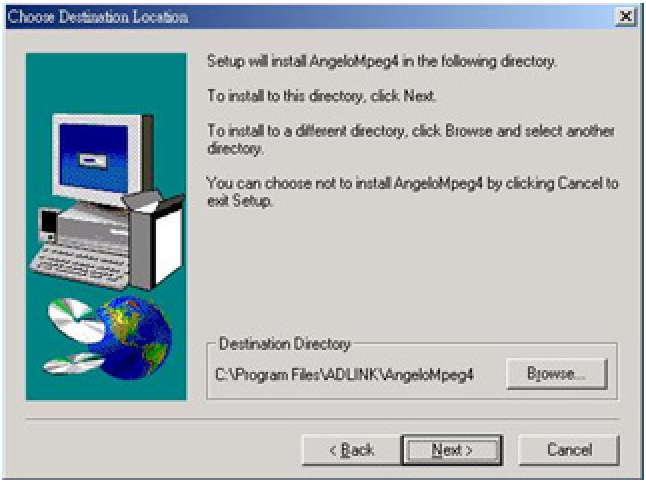
→ GEME-2000 (please refer to \GEME\GEME-2000)

→ GEME-3000 (please refer to \GEME\GEME-3000) include on-boad Bt878 Drive

MPC-7664 | PMC-3534
MPC-8104 | PMC-3544
MPC-8164 | PMC-7852
MPC-8372 | ● Software-MPEG4
PMC-RTV21 | PMC-8611/PMC-8615 (pls refer to \GEME\PMC-8615)

4. The driver will begin installing.

5. Click Next until the driver installs completely.



Setup will install AngeloMpeg4 in the following directory.

To install to this directory, click Next.

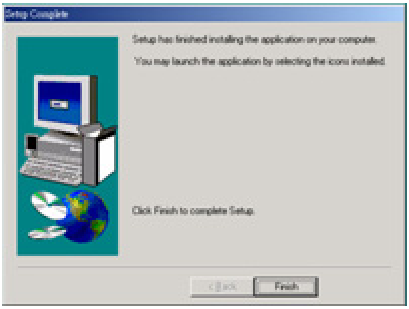
To install to a different directory, click Browse and select another directory.

You can choose not to install AngeloMpeg4 by clicking Cancel to exit Setup.

Destination Directory

C:\Program Files\ADLINK\AngeloMpeg4

6. Click Finish and restart the system.

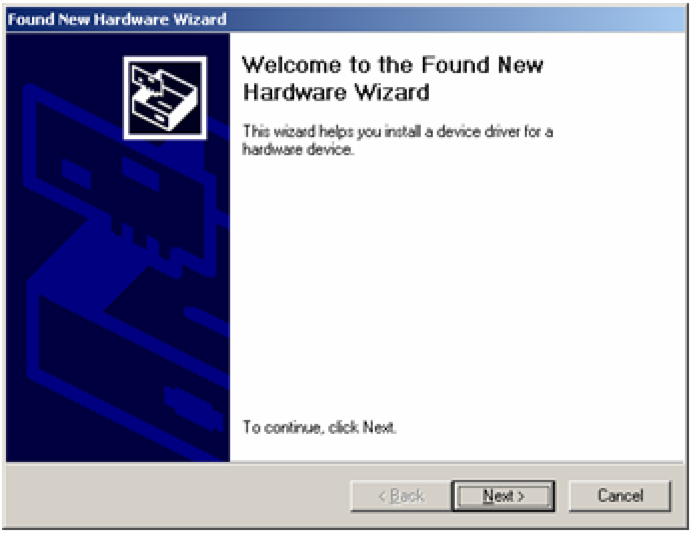


Setup has finished installing the application on your computer.

You may launch the application by selecting the icons installed.

Click Finish to complete Setup.

7. The Found New Hardware Wizard window appears after system restarts. Click Next and follow these steps to complete the new hardware wizard.

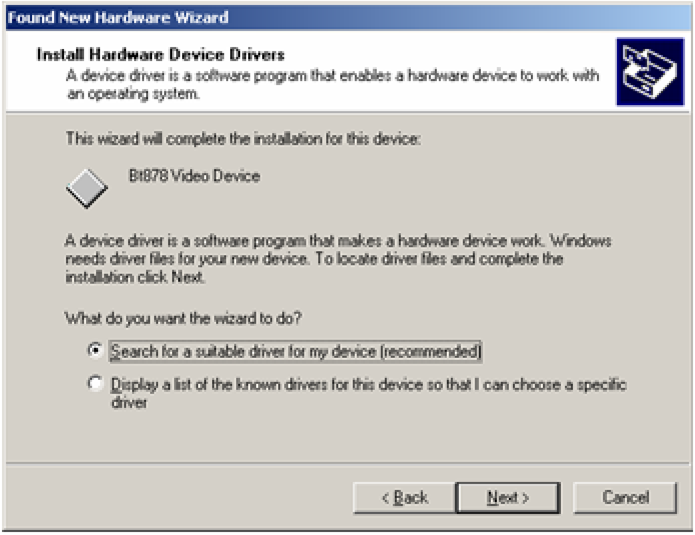


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install a device driver for a hardware device.

To continue, click Next.



Found New Hardware Wizard

Install Hardware Device Drivers

A device driver is a software program that enables a hardware device to work with an operating system.

This wizard will complete the installation for this device:

Bt878 Video Device

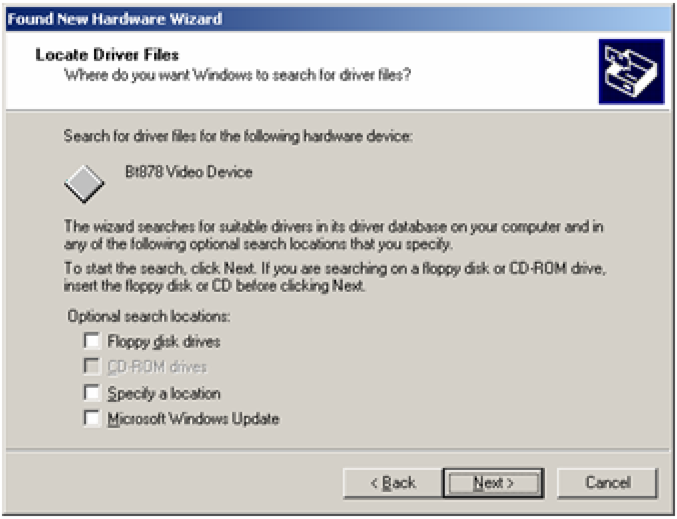
A device driver is a software program that makes a hardware device work. Windows needs driver files for your new device. To locate driver files and complete the installation click Next.

What do you want the wizard to do?

● Search for a suitable driver for my device (recommended)

○ Display a list of the known drivers for this device so that I can choose a specific driver

Click Next.



Found New Hardware Wizard

Locate Driver Files

Where do you want Windows to search for driver files?

Search for driver files for the following hardware device:

Bt878 Video Device

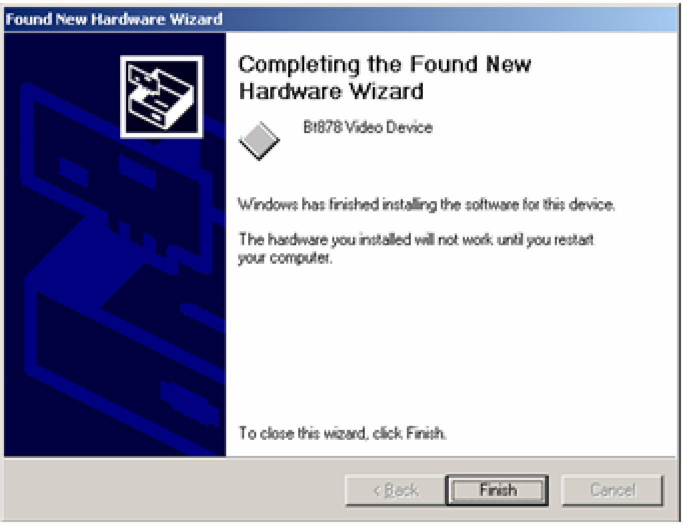
The wizard searches for suitable drivers in its driver database on your computer and in any of the following optional search locations that you specify.

To start the search, click Next. If you are searching on a floppy disk or CD-ROM drive, insert the floppy disk or CD before clicking Next.

Optional search locations:

- □ Floppy disk drives
- □ CD-ROM drives
- □ Specify a location
- □ Microsoft Windows Update

Click Next.



Found New Hardware Wizard

Completing the Found New Hardware Wizard

Bt878 Video Device

Windows has finished installing the software for this device.

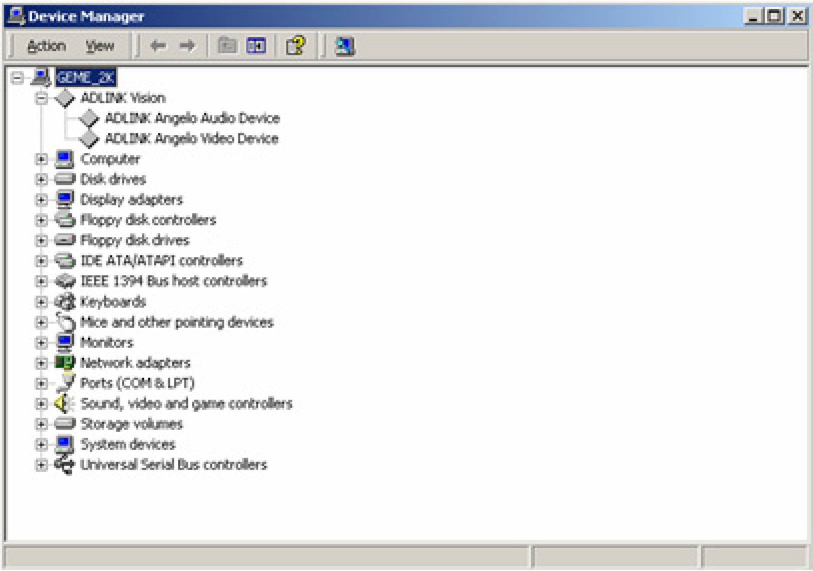
The hardware you installed will not work until you restart your computer.

To close this wizard, click Finish.

Click Finish.

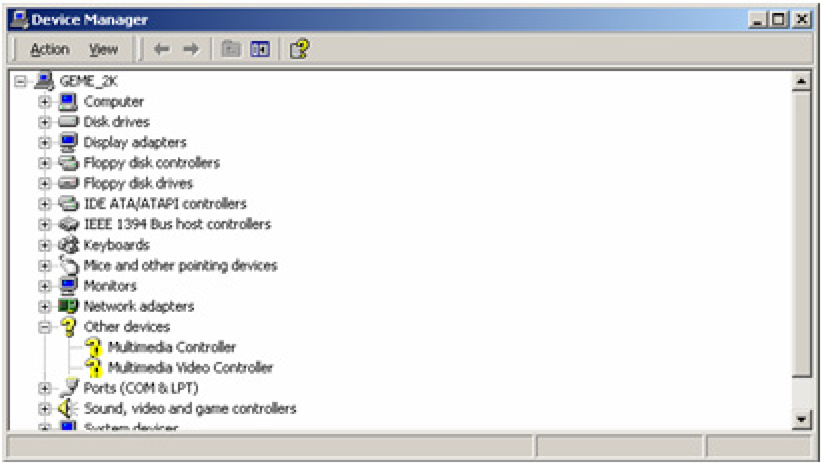
8. Another Found New Hardware Wizard window appears when you finish the wizard. Repeat Step 7 until you finish installing all new hardware.

9. Go to the Device Manager and check to see that the "ADLINK Angelo Audio Device" and "ADLINK Angelo Video Device" are installed as shown.

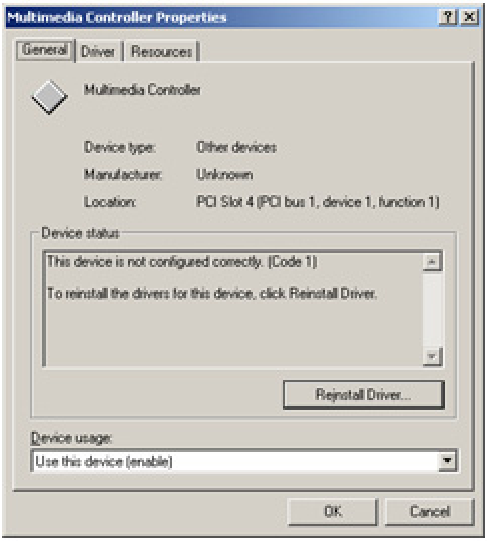


- GEME_2K
  - ADLINK Vision
    - ADLINK Angelo Audio Device
    - ADLINK Angelo Video Device
  - Computer
  - Disk drives
  - Display adapters
  - Floppy disk controllers
  - Floppy disk drives
  - IDE ATA/ATAPI controllers
  - IEEE 1394 Bus host controllers
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Ports (COM & LPT)
  - Sound, video and game controllers
  - Storage volumes
  - System devices
  - Universal Serial Bus controllers

10. If you see a yellow question mark appearing in front of a driver name, you will need to setup the driver manually.



11. Right click on the driver named Multimedia Controller, which is an audio device. Then select Properties in the popup menu. Follow these steps to complete the driver installation.



Multimedia Controller

Device type: Other devices

Manufacturer: Unknown

Location: PCI Slot 4 (PCI bus 1, device 1, function 1)

Device status

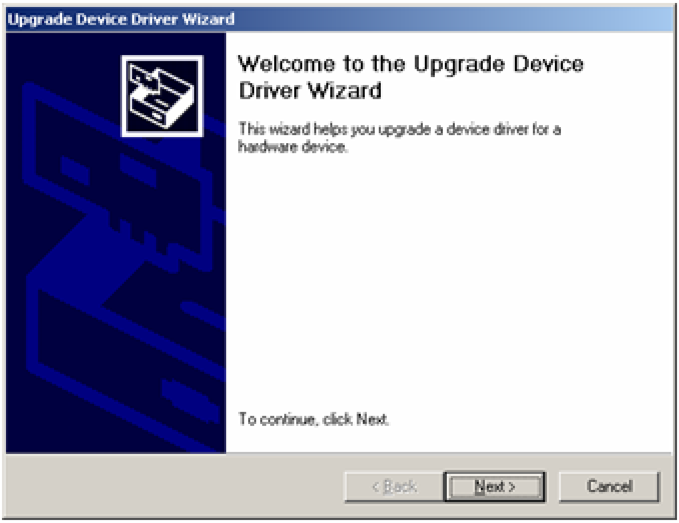
This device is not configured correctly. (Code 1)

To reinstall the drivers for this device, click Reinstall Driver.

Device usage:

Use this device (enable)

Click Reinstall Driver.



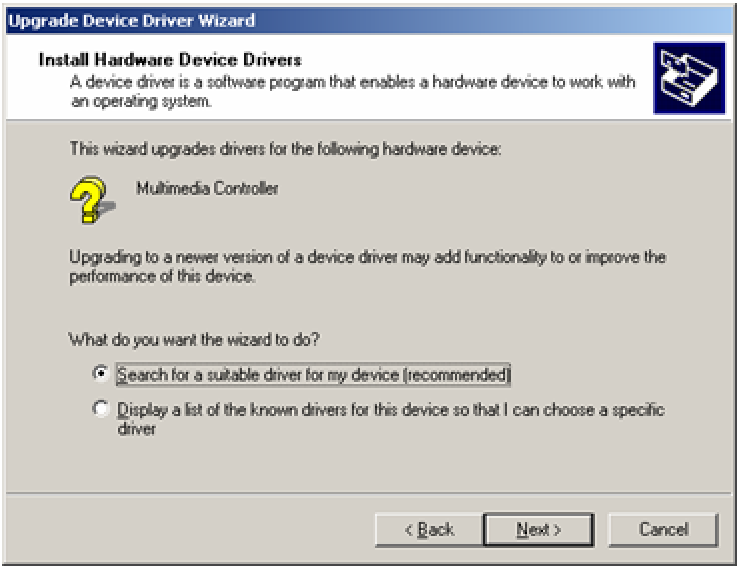
Upgrade Device Driver Wizard

Welcome to the Upgrade Device Driver Wizard

This wizard helps you upgrade a device driver for a hardware device.

To continue, click Next.

Click Next.



Upgrade Device Driver Wizard

Install Hardware Device Drivers

A device driver is a software program that enables a hardware device to work with an operating system.

This wizard upgrades drivers for the following hardware device:

Multimedia Controller

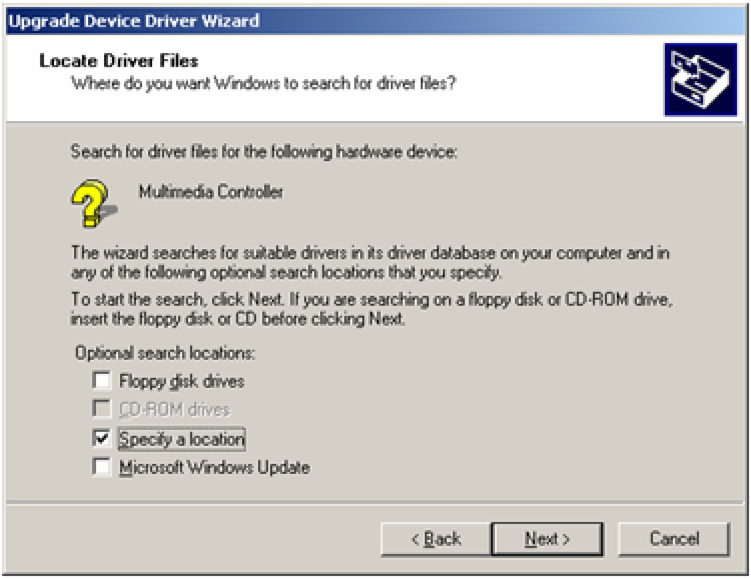
Upgrading to a newer version of a device driver may add functionality to or improve the performance of this device.

What do you want the wizard to do?

● Search for a suitable driver for my device (recommended)

○ Display a list of the known drivers for this device so that I can choose a specific driver

Click Next.



Upgrade Device Driver Wizard

Locate Driver Files

Where do you want Windows to search for driver files?

Search for driver files for the following hardware device:

Multimedia Controller

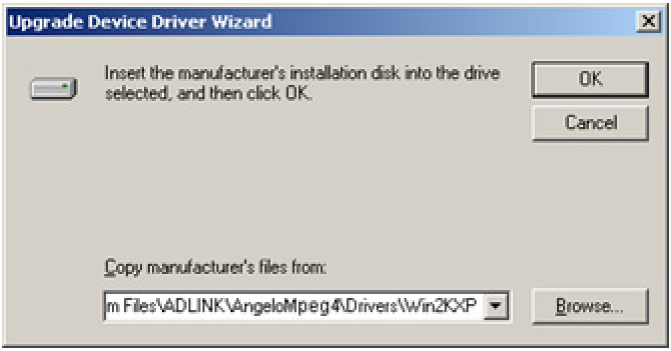
The wizard searches for suitable drivers in its driver database on your computer and in any of the following optional search locations that you specify.

To start the search, click Next. If you are searching on a floppy disk or CD-ROM drive, insert the floppy disk or CD before clicking Next.

Optional search locations:

- □ Floppy disk drives
- □ CD-ROM drives
- ☑ Specify a location
- □ Microsoft Windows Update

Check Specify a location and then click Next.



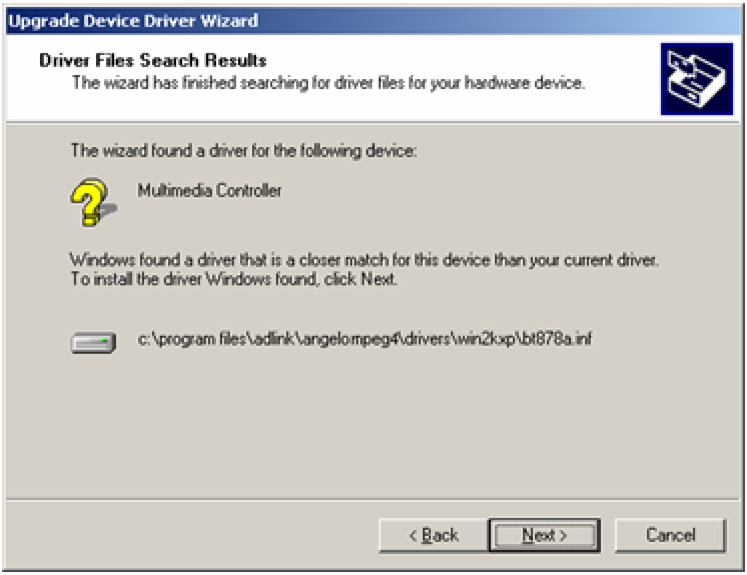
Upgrade Device Driver Wizard

Insert the manufacturer's installation disk into the drive selected, and then click OK.

Copy manufacturer's files from:

m Files\ADLINK\AngeloMpeg4\Drivers\Win2KXP

In the Copy manufacture's files from: text box, input the location of the driver installed in Step 5 (for example, '`C:\Program Files\ADLINK\AngeloMpeg4\Drivers\Win2KXP`'). Then click OK.



Upgrade Device Driver Wizard

Driver Files Search Results

The wizard has finished searching for driver files for your hardware device.

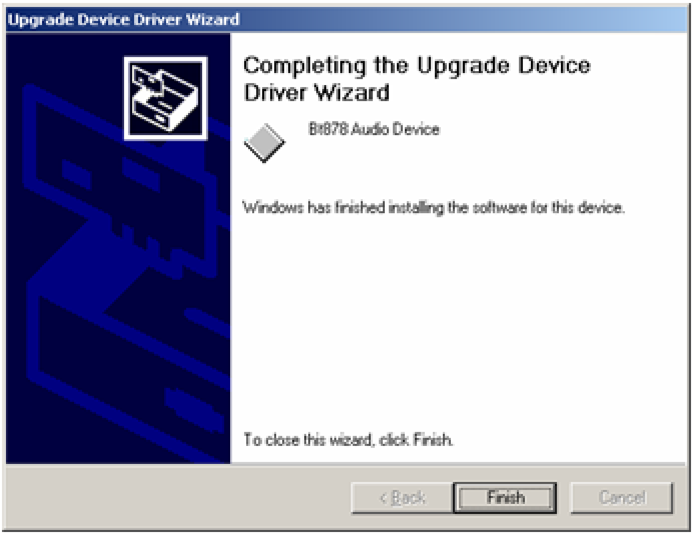
The wizard found a driver for the following device:

Multimedia Controller

Windows found a driver that is a closer match for this device than your current driver. To install the driver Windows found, click Next.

c:\program files\adlink\angelompeg4\drivers\win2kxp\bt878a.inf

Click Next.



Upgrade Device Driver Wizard

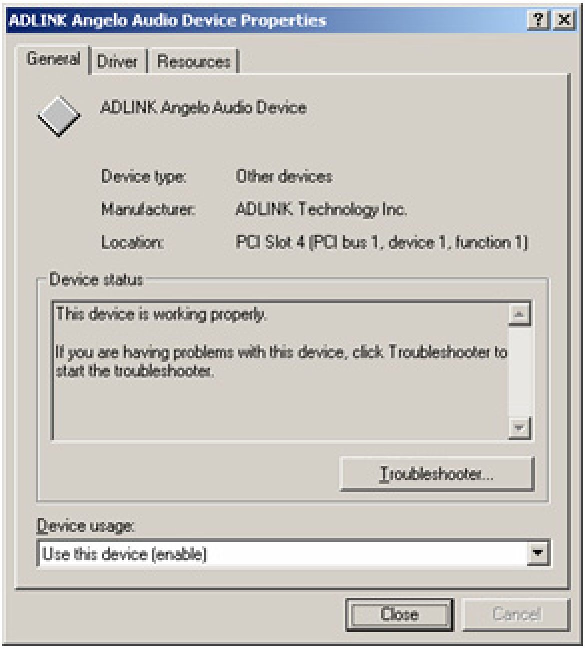
Completing the Upgrade Device Driver Wizard

Bt878 Audio Device

Windows has finished installing the software for this device.

To close this wizard, click Finish.

Click Finish to complete this wizard.



ADLINK Angelo Audio Device

Device type: Other devices

Manufacturer: ADLINK Technology Inc.

Location: PCI Slot 4 (PCI bus 1, device 1, function 1)

Device status

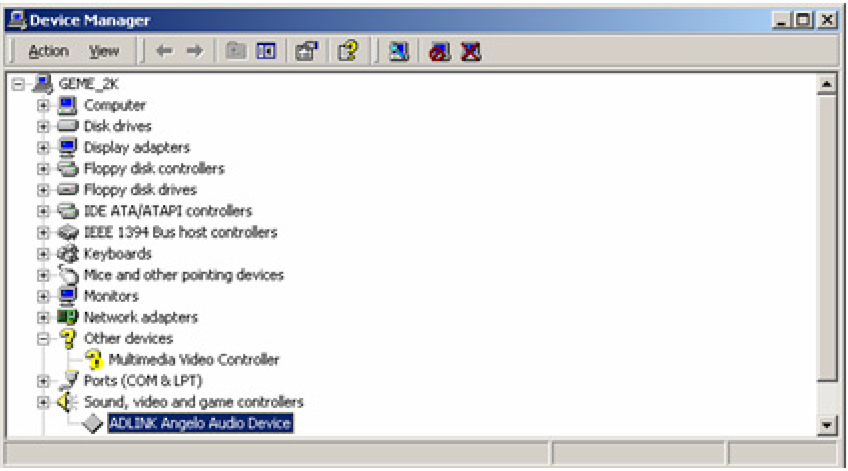
This device is working properly.

If you are having problems with this device, click Troubleshooter to start the troubleshooter.

Device usage:

Use this device (enable)

Congratulations! This device is working properly.



- GEME_2K
  - Computer
  - Disk drives
  - Display adapters
  - Floppy disk controllers
  - Floppy disk drives
  - IDE ATA/ATAPI controllers
  - IEEE 1394 Bus host controllers
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Other devices
    - Multimedia Video Controller
  - Ports (COM & LPT)
  - Sound, video and game controllers
    - ADLINK Angelo Audio Device

The yellow question mark will disappear.

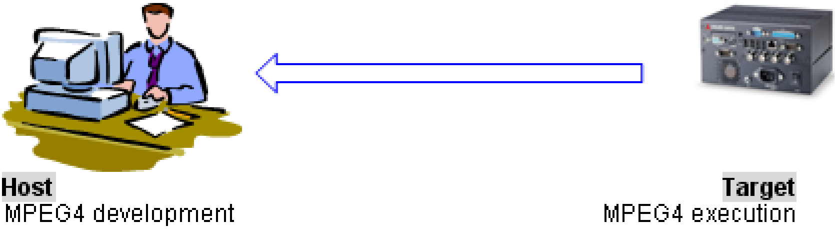
12. Right click on the driver named Multimedia Video Controller, which is a video device. Repeat Step 11 to install the driver.

Note: If the system prompts you to restart computer before you have completed installing all devices, please select No. After all drivers have been installed, restart the computer to allow the drivers to take effect.

Driver Installation for Windows XP Embedded

For MPEG4 program development, GEME can take the role of Target. The user then takes the role of Host to develop his/her own MPEG4 program.

ADLINK will pre-install the MPEG4 driver when the GEME-VM3000 is ordered with a Windows XP Embedded license. The user then needs to install the MPEG4 driver in the Host environment according to the instructions in Section C.2.1.



6.3 ViewCreator Utility

Once hardware installation is complete, ensure that the system is correctly configured before running the ViewCreator utility. This chapter outlines how to set up a vision system and manually control Angelo series cards to verify correct operation. ViewCreator provides a simple yet powerful means to setup, configure, test, and debug the vision system.

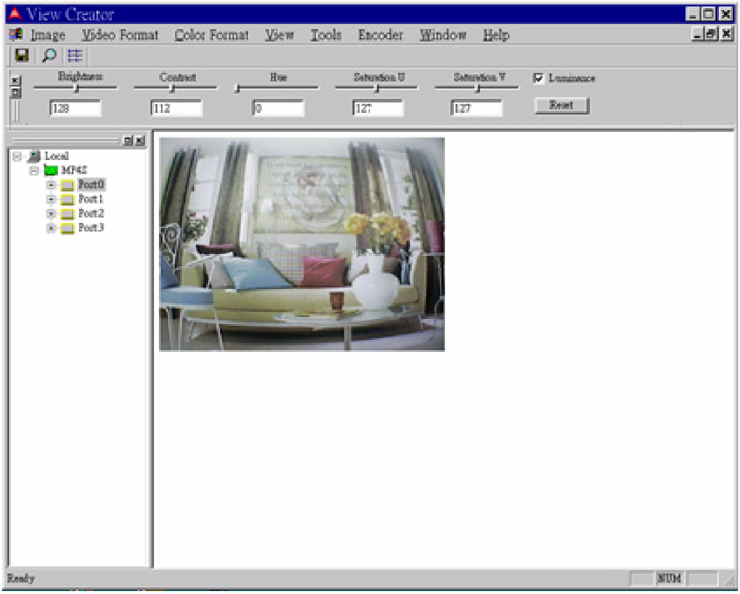
Note: ViewCreator is only available for Windows 98/NT/2k/XP with a recommended screen resolution of 800x600 or higher.

Overview

ViewCreator offers the following features:

- 32-bit operation under Windows 98/2000/XP
- Angelo series cards access and configuration
- Video picture adjustments
- MPEG4 video encoding
- Recording (AVI video format)
- Direct access to general purpose I/Os
- FULL, CIF, or QCIF image size, 2x2 or 4x4 display
- Software triggering

Component Description



Brightness 128
Contrast 112
Hue 0
Saturation U 127
Saturation V 127
☑ Luminance

Tree Browser

The Tree Browser window lists the PCI-MP4S cards and video ports available at the local computer.

Image View

The Image View window displays Full, CIF, and QCIF size images and image effects.

Control Panel

The control panel allows for making video adjustments, including brightness, hue, contrast, etc.

Operation Theory

ViewCreator provides many functions for the Angelo series card as described below.

MPEG4 Encoding

Single channel display

- Click a video Port icon in the Tree Browser window. A video frame will appear in the Image View window.
- Select Encoder->Encode in menu bar to bring up the Encoder Setting dialog box, then click the start button.

Note:

1. View Creator supports only one channel CIF video encoding. Ensure there is only one channel, CIF image on the screen.
2. Execute the decoder sample program in Program files->ADLINK->AngeloMPEG4->Samples to connect to the encoder (IP:127.0.0.1 for local computer)

Video image configuration

Video format

Click Format in the menu bar to select the format of the video camera. The supported video formats are NTSC, EIA, PAL, and CCIR.

Color format

The default color format setting in ViewCreator is RGB24. The color format of the application can be changed.

Video size

Click View in the menu bar and select the image size required. The supported video sizes are listed below:

- ▷ **FULL:** 640x480 for NTSC, EIA and 768x576 for PAL, CCIR
- ▷ **CIF:** 320x240 for NTSC, EIA and 384x288 for PAL, CCIR
- ▷ **QCF:** 160x120 for NTSC, EIA and 192x144 for PAL, CCIR

Video adjustments

Hue

Click and hold the left mouse button on the Hue slider of the Control Panel and drag the cursor to change its value. Values range from 0 to 255.

Contrast

Click and hold the left mouse button on the Contrast slider of the Control Panel and drag the cursor to change its value. Values range from 0 to 255.

Brightness

Click and hold the left mouse button on the Brightness slider of the Control Panel and drag the cursor to change its value. Values range from 0 to 255.

Save image file

This function can only be used in single channel display mode (select a video Port icon in the Tree Browser window).

TIF

Click Image in the menu bar and select **Save As** to bring up the Save As dialog box. Select the file location, TIF file format, enter the file name, and click **OK**.

BMP

Click **Image** in the menu bar and select **Save As** to bring up the Save As dialog box. Select the file location, BMP file format, enter the file name, and click **OK**.

Special image effect

Border

- Check the **Border** check box in the Control Panel. A red dashed border will appear around the image.
- Drag the red line to resize the border. Only the image within the border will refresh.

Cross Line

Check the Cross Line check box in the Control Panel. A cross-hair will appear in the center of the rectangle.

Tools

GPIO and LED

- Click Tool in the menu bar and select **GPIO & LED** to bring up the GPIO dialog box. Select the port to access and select the digital output value. Click either the **write** or **read** button to write/read to/from the digital I/O ports.
- LED status is only supported with the cPCI Angelo series card.

EEPROM

- Click **Tool** in the menu bar and select EEPROM to bring up the EEPROM dialog box. Select the card you wish to access, enter the offset and output values, then click the **Write** button to write the value into the EEPROM. Enter the offset value and click the **Read** button to read the value from the EEPROM.
- Valid offset values are between 0 and 127. Valid output values are between 0 and 255. The value in the EEPROM will not be erased when the system is powered off.

Software trigger

- Click **Tool** in the menu bar and select **Software Trigger** to bring up the Trigger dialog box. Select the card to access and set the interval of the trigger pulse output. Check the ports you want to trigger simultaneously, and click the **Trigger** button.
- The one shot pulse output voltage goes high (from 0V to 5V).

6.4 Function Library

This chapter describes the API for Mpeg4 encode and decode. Users can use these functions to develop application programs under Visual C++, Visual Basic, C++ Builder, and Delphi.

List of Functions

Category	Function
Encode	AngeloMPEG4_Encode_Initial(Encoder_Index, Local_Address, Quality , Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)
	AngeloMPEG4_Encode_InitialEx(Encoder_Index, Local_Address, Bitrate, frame_rate, Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)
	AngeloMPEG4_Encode_Set_Callback(Encoder_Index, CallBackProc)
	AngeloMPEG4_Encode_Start(Encoder_Index)
	AngeloMPEG4_Encode_Stop(Encoder_Index)
	AngeloMPEG4_Encode_Close(Encoder_Index)
	AngeloMPEG4_Encode_Save_File_Start(Encoder_Index, n_file_name, interval_second, format)
	AngeloMPEG4_Encode_Save_File_Stop(Encoder_Index)
	AngeloMPEG4_Encode_Create_Directory(Encoder_Index, Dir)
	AngeloMPEG4_Encode_Set_Motion_Detection(Encoder_Index, Area, enable, Threshold, interval, action, X_Start, Y_Start, Width, Height)

Table 6-2: List of Functions

Category	Function
Decode	AngeloMPEG4_Decode_Set_Callback(Decoder_Index, CallBackProc)
	AngeloMPEG4_Decode_Connect(Decoder_Index,Encoder_IP, Encoder_Index)
	AngeloMPEG4_Decode_Disconnect(Decoder_Index)
	AngeloMPEG4_Decode_Set_Image_Config(Decoder_Index, ConfigIndex , Value)
	AngeloMPEG4_Decode_Set_Motion_Detection(Decoder_Index, Area, enable, Threshold, interval,action, X_Start, Y_Start, Width, Height)
	AngeloMPEG4_Decode_Get_Config(Decoder_Index,iWidth, iHeight, video_format, color_format,Bitrate, frame_rate)
	AngeloMPEG4_Decode_Start(Decoder_Index)
	AngeloMPEG4_Decode_Stop(Decoder_Index)
	AAngeloMPEG4_Decode_Get_FlowRate(Decoder_Index, Byte_Second)
	AngeloMPEG4_Decode_ReInitialEx(Decoder_Index, Bitrate, frame_rate, Angelo_Video_Format)
	AngeloMPEG4_Decode_ReInitial(Decoder_Index, Quality, Angelo_Video_Format)
	AngeloMPEG4_Decode_Save_File_Start(Decoder_Index, n_file_name, interval_second, format)
	AngeloMPEG4_Decode_Save_File_Stop(Decoder_Index)
	AngeloMPEG4_Decode_File(Decoder_Index, file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame,Time_Seconds)
	AngeloMPEG4_Decode_File_Start(Decoder_Index, Mode)
	AngeloMPEG4_Decode_File_Set_Position(Decoder_Index, Frame_Index)
	AngeloMPEG4_Decode_File_Pause(Decoder_Index)
	AngeloMPEG4_Decode_File_Continue(Decoder_Index)
	AngeloMPEG4_Decode_File_Get_Position(Decoder_Index, Cur_Frame_Index)
	AngeloMPEG4_AVI_2_M4V(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds)
	AngeloMPEG4_M4V_2_AVI(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds)
System	AngeloMPEG4_Get_Version(Mpeg4_DLLVersion, AngeloRTV_DLLVersion, Reserved)

Table 6-2: List of Functions

Encode Functions

@ Name

`AngeloMPEG4_Encode_Initial(Encoder_Index, Local_Address, Quality , Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format)` : Initialize the encoder.

AngeloMPEG4_Encode_InitialEx(Encoder_Index, Local_Address, Bitrate, frame_rate, Angelo_PortNo, Angelo_ChannelNo, Angelo_Color_Format, Angelo_Video_Format): Initialize the encoder for advanced.

AngeloMPEG4_Encode_Set_Callback(Encoder_Index, CallBackProc): Set up the callback function for encoder.

AngeloMPEG4_Encode_Start(Encoder_Index): Start to grab image and encode.

AngeloMPEG4_Encode_Stop(Encoder_Index): Stop grabbing image and encoding.

AngeloMPEG4_Encode_Close(Encoder_Index): Close the encoder and network transmission.

AngeloMPEG4_Encode_Save_File_Start(Encoder_Index, n_file_name, interval_second, format): Start to save compressed file in encode site.

AngeloMPEG4_Encode_Save_File_Stop(Encoder_Index): Stop saving compressed file in encode site.

AngeloMPEG4_Encode_Create_Directory(Encoder_Index, Dir): Create a new folder on the encode site.

AngeloMPEG4_Encode_Set_Motion_Detection(Encoder_Index, Area, enable, Threshold, interval, action, X_Start, Y_Start, Width, Height): Set the motion detection criteria, and action when motion occurs on the encode site.

@ Description

AngeloMPEG4_Encode_Initial:

This function initializes the video encoder. Its library supports 16 video encoders with the video source coming from Angelo_PortNo and Angelo_ChannelNo in the Angelo cards. Quality, and Angelo_Color_Format are parameters for encoder setting.

AngeloMPEG4_Encode_InitialEx:

This function initializes the video encoder. Its library supports 16 video encoders with the video source coming from Angelo_PortNo and Angelo_ChannelNo in the Angelo cards. Bitrate, frame_rate, and Angelo_Color_Format are parameters for encoder setting.

`AngeloMPEG4_Encode_Set_Callback:`

This function establishes a notification mechanism between function library and user process. Callback function is application-defined. The user passes the function pointer to function library by calling this function.

`AngeloMPEG4_Encode_Start:`

This function restarts encoding the video image when the encoder is paused.

`AngeloMPEG4_Encode_Stop:`

This function pauses encoding of the video image.

`AngeloMPEG4_Encode_Close:`

This function releases the resources of the encoder for the specified channel.

`AngeloMPEG4_Encode_Save_File_Start:`

Use this function to save the encoded image into an ".avi" or ".m4v" video file. The ".avi" file is the standard video format, and ".m4v" is only accessible in this function library.

Note:

1. Do not add a file extension to the file name.
2. User must install the XVID Codec in our setup disk in order to play ".avi" file in MS Media Player.

`AngeloMPEG4_Encode_Save_File_Stop:`

Use this function to stop saving the video file. In general, the video file will close automatically after the "Interval" parameter in AngeloMPEG4_Encode_Save_File_Start.

`AngeloMPEG4_Encode_Create_Directory:`

This function is used to create a new directory for saving a video file. The "filename" parameter in

AngeloMPEG4_Encode_Save_File_Start contains the file path name.

AngeloMPEG4_Encode_Set_Motion_Detection:

Use this function to configure the motion detection criteria and the action when motion occurs at the encoding site.

AngeloMPEG4_Encode_Initial–

AngeloMPEG4_Encode_InitialEx–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Encode_Initial(int Encoder_Index,
     char* Local_Address, int Quality ,int
     Angelo_PortNo, int Angelo_ChannelNo, int
     Angelo_Color_Format, int
     Angelo_Video_Format)
int AngeloMPEG4_Encode_InitialEx(int
     Encoder_Index, char* Local_Address, int
     Bitrate, int frame_rate, int Angelo_PortNo,
     int Angelo_ChannelNo, int
     Angelo_Color_Format, int
     Angelo_Video_Format);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Initial(ByVal Encoder_Index As
     Long, ByVal Local_address As String, ByVal
     Quality As Long, ByVal Angelo_PortNo As
     Long, ByVal Angelo_ChannelNo As Long, ByVal
     Angelo_Color_Format As Long, ByVal
     Angelo_Video_Format As Long) As Long
AngeloMPEG4_Encode_InitialEx (ByVal Encoder_Index
     As Long, ByVal Local_address As String,
     ByVal Bitrate As Long, ByVal frame_rate As
     Long, ByVal Angelo_PortNo As Long, ByVal
     Angelo_ChannelNo As Long, ByVal
     Angelo_Color_Format As Long, ByVal
     Angelo_Video_Format As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Initial(Encoder_Index:Longint
        ; Local_Address:String; Quality:Longint;
        Angelo_PortNo:Longint;
        Angelo_ChannelNo:Longint;
        Angelo_Color_Format:Longint;
        Angelo_Video_Format:Longint):Longint;
AngeloMPEG4_Encode_InitialEx(Encoder_Index:Longi
        nt; Local_Address:String; Bitrate:Longint;
        frame_rate:Longint; Angelo_PortNo:Longint;
        Angelo_ChannelNo:Longint;
        Angelo_Color_Format:Longint;
        Angelo_Video_Format:Longint):Longint;
```

@ Argument

`Encoder_Index`: Indicates the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

`Local_Address`: Indicates the IP Address at the encoding site. Set 0, NULL or nil for default setting.

`Quality Index`:

Quality Level	Value	NTSC		
		4CIF (640 x 480)	CIF (320 x 240)	QCIF (160 x 120)
Lowest	-2	Bit-rate = 400000*4 Frame rate = 5	Bit-rate = 400000 Frame rate = 5	Bit-rate = 400000/4 Frame rate = 5
Low	-1	Bit-rate = 480000*4 Frame rate = 10	Bit-rate = 480000 Frame rate = 10	Bit-rate = 480000/4 Frame rate = 10
Normal	0	Bit-rate = 560000*4 Frame rate =15	Bit-rate = 560000 Frame rate =15	Bit-rate = 560000/4 Frame rate =15
High	1	Bit-rate = 560000*4 Frame rate = 30	Bit-rate = 560000 Frame rate = 30	Bit-rate = 560000/4 Frame rate = 30
Highest	2	Bit-rate = 1024000*4 Frame rate = 30	Bit-rate = 1024000 Frame rate = 30	Bit-rate = 1024000/4 Frame rate = 30
Quality Level	**Value**	**PAL**		
		4CIF (768 x 576)	**CIF (384 x 288)**	**QCIF (192 x 144)**
Lowest	-2	Bit-rate = 400000*4 Frame rate = 4	Bit-rate = 400000 Frame rate = 4	Bit-rate = 400000/4 Frame rate = 4

Table 6-3: Quality Index

Low	-1	Bit-rate = 480000*4 Frame rate = 8	Bit-rate = 480000 Frame rate = 8	Bit-rate = 480000/4 Frame rate = 8
Normal	0	Bit-rate = 560000*4 Frame rate =12	Bit-rate = 560000 Frame rate =12	Bit-rate = 560000/4 Frame rate =12
High	1	Bit-rate = 560000*4 Frame rate = 25	Bit-rate = 560000 Frame rate = 25	Bit-rate = 560000/4 Frame rate = 25
Highest	2	Bit-rate = 1024000*4 Frame rate = 25	Bit-rate = 1024000 Frame rate = 25	Bit-rate = 1024000/4 Frame rate = 25

Table 6-3: Quality Index

`Bitrate:` Indicates the number of bits per second.

`frame_rate:` Indicates the number of frames that the MPEG4 encoder will encode per second. The range of the frame_rate is 1 – 30.

`Angelo_PortNo:` The port number is the zero index of the Angelo series card. For example, if there are two PCI-RTV-24 Angelo cards (card 0, card 1) in the system, and each PCI-RTV-24 has four ports, the first port of card 0 is “0”, and the first port of card 1 is “4.”

`Angelo_ChannelNo:` Indicates the channel index of the port described above. There are four channels per port and the first channel index is 0.

`Angelo_Color_Format:` RGB24= 3

`Angelo_Video_Format:`

0: Full NTSC, with image size 640*480

1: Full PAL, with image size 768*576

2: CIF NTSC, with image size 320*240

3: CIF PAL, with image size 384*288

4: QCIF NTSC, with image size 160*120

5: QCIF PAL, with image size 192*144

@ Return Code

@ Example

<VC/BCB >

```
    int Result;
```

```
int Encoder_Index = 0;
int Quality = 0;
int Angelo_PortNo = 0;
int Angelo_ChannelNo = 0;
int Angelo_Color_Format = 3; //RGB24
int Angelo_Video_Format = 2; //CIF NTSC
int Bitrate = 480000;
int frame_rate = 15;
Result = AngeloMPEG4_Encode_Initial
      (Encoder_Index, Quality, Angelo_PortNo,
      Angelo_ChannelNo, Angelo_Color_Format,
      Angelo_Video_Format);
Result = AngeloMPEG4_Encode_InitialEx
      (Encoder_Index, Bitrate, frame_rate,
      Angelo_PortNo, Angelo_ChannelNo,
      Angelo_Color_Format, Angelo_Video_Format);
```

< Visual Basic >

```
Dim result As Long
Dim Encoder_Index As Long, Quality As Long,
      Angelo_PortNo As Long, Angelo_ChannelNo As
      Long, Angelo_Color_Format As Long,
      Angelo_Video_Format As Long, Bitrate As
      Long, frame_rate As Long
Encoder_Index =  0
Quality = 0
Angelo_PortNo = 0
Angelo_ChannelNo = 0
Angelo_Color_Format = 3 "RGB24
Angelo_Video_Format = 2 "CIF NTSC
Bitrate = 480000
frame_rate = 15
Result = AngeloMPEG4_Encode_Initial
      (Encoder_Index, Quality, Angelo_PortNo,
      Angelo_ChannelNo, Angelo_Color_Format,
      Angelo_Video_Format)
Result = AngeloMPEG4_Encode_InitialEx
      (Encoder_Index, Bitrate, frame_rate,
      Angelo_PortNo, Angelo_ChannelNo,
      Angelo_Color_Format, Angelo_Video_Format)
```

< Delphi >

```
Var
Encoder_Index, Result: Longint;
Quality: Longint;
Bitrate, frame_rate: Longint;
Angelo_PortNo, Angelo_ChannelNo: Longint;
Angelo_Color_Format, Angelo_Video_Format:
     Longint;
begin
Encoder_Index:= 0;
Quality := 0; // Normal Quality
Bitrate := 480000;
frame_rate := 15;
Angelo_PortNo := 0;
Angelo_ChannelNo := 0;
Angelo_Color_Format := 3; // RGB24
Angelo_Video_Format := 2; // CIF, NTSC
Result :=
     AngeloMPEG4_Encode_Initial(Encoder_Index,
     Quality, Angelo_PortNo, Angelo_ChannelNo,
     Angelo_Color_Format, Angelo_Video_Format);
Result :=
     AngeloMPEG4_Encode_InitialEx(Encoder_Index,
     Bitrate, frame_rate, Angelo_PortNo,
     Angelo_ChannelNo, Angelo_Color_Format,
     Angelo_Video_Format);
end;
```

AngeloMPEG4_Encode_Set_Callback–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Callback(int
     Encoder_Index, void ( __stdcall
     *CallBackProc)(int Encoder_Index,long
     int_status,param_str* param_struct));
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Callback (ByVal
     Encoder_Index As Long, ByVal
     Encode_CallBackProcas As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Callback(Encoder_Index:Lo
      ngint;
      lpEncodeCallBackProc:EncodeCallBackProc):Lo
      ngint;
```

@ Argument

`Encoder_Index`: Indicates the channel Index for the MPEG4 encoder. The range of channels is 0 – 15.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int Encoder_Index = 0;
void __stdcall Encode_Callback(int Encoder_Index,
      long int_status, param_str *param_struct)
{
      if(int_status & 0x01 ==1) //Image Ready
{
      }
      if(int_status >> 4 & 0x01 ==1) //Motion
      Dection
{
      }
}
Result =
      AngeloMPEG4_Encode_Set_Callback(Encoder_Ind
      ex, Encode_Callback);
```

< Visual Basic >

```
Dim Encoder_Index As Long, Result As Long
Public Sub encode_callback(ByVal Encoder_Index As
     Long, ByVal int_status As Long, param_str As
     param_struct)
    Select Case (int_status)
    Case 1: "preview
Case 16: " motion detection
End Select
End Sub
Channel =0
Result =
     AngeloMPEG4_Encode_Set_Callback(Encoder_Ind
     ex, Encode_Callback)
```

< Delphi >

```
procedure Encode_Callback
     (Encoder_Index:Longint;int_status:Longint;v
     ar param_struct:param_str);stdcall
var
    {* add your var here *}
begin
    case int_status of
      1: begin  {********* Image Ready *********}
      end;
     2: begin  {********* Set Image Config Event
     *********}
      end;
      4: begin  {********* Connected Event
     *********}
      end;
      8: begin  {********* Disconnect Event
     *********}
      end;
     16: begin  {********* Motion Detection Event
     *********}
      end;
    end;  // end case int_status of
end;
// Main Code
var
Encoder_Index, Result: Longint;
begin
```

```
Encoder_Index:= 0;
Result :=
     AngeloMPEG4_Encode_Set_Callback(Encoder_Ind
     ex, Encode_Callback);
end;
```

AngeloMPEG4_Encode_Start–

AngeloMPEG4_Encode_Stop–

AngeloMPEG4_Encode_Close–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Encode_Start(int Encoder_Index);
int AngeloMPEG4_Encode_Stop(int Encoder_Index);
int AngeloMPEG4_Encode_Close(int Encoder_Index);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Start (ByVal Encoder_Index As
     Long) As Long
AngeloMPEG4_Encode_Stop (ByVal Encoder_Index As
     Long) As Long
AngeloMPEG4_Encode_Close (ByVal Encoder_Index As
     Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Start(Encoder_Index:Longint):
     Longint;
AngeloMPEG4_Encode_Stop(Encoder_Index:Longint):L
     ongint;
AngeloMPEG4_Encode_Close(Encoder_Index:Longint):
     Longint;
```

@ Argument

`Channel:` Indicates the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

@ Return Code

@ Example

< VC/BCB >

```
 int Result;
```

```
int Encoder_Index = 0;

Result = AngeloMPEG4_Encode_Start(Encoder_Index);
Result = AngeloMPEG4_Encode_Stop(Encoder_Index);
Result = AngeloMPEG4_Encode_Close(Encoder_Index);
```

< Visual Basic >

```
Dim Result As Long, Encoder_Index As Long

Encoder_Index = 0

Result = AngeloMPEG4_Encode_Start(Encoder_Index)
Result = AngeloMPEG4_Encode_Stop(Encoder_Index)
Result = AngeloMPEG4_Encode_Close(Encoder_Index)
```

< Delphi >

```
var
Encoder_Index, Result: Longing;
begin
Result := AngeloMPEG4_Encode_Stop(Encoder_Index);
     // pause the encoder
Result :=
     AngeloMPEG4_Encode_Start(Encoder_Index); //
     restart the encoder
// close the Encoder
Result := AngeloMPEG4_Encode_Stop(Encoder_Index);
Result :=
     AngeloMPEG4_Encode_Close(Encoder_Index);
end;
```

AngeloMPEG4_Encode_Save_File_Start–

AngeloMPEG4_Encode_Save_File_Stop–

AngeloMPEG4_Encode_Create_Directory–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int  AngeloMPEG4_Encode_Save_File_Start(int
     Encoder_Index, char* n_file_name, long
     interval_second, long format);
int  AngeloMPEG4_Encode_Save_File_Stop(int
     Encoder_Index);
int AngeloMPEG4_Encode_Create_Directory(int
     Encoder_Index, char* Dir);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Save_File_Start (ByVal
     Encoder_Index As Long, ByVal n_file_name As
     String, ByVal interval_second As Long, ByVal
     format As Long) As Long
AngeloMPEG4_Encode_Save_File_Stop (ByVal
     Encoder_Index As Long) As Long
AngeloMPEG4_Encode_Create_Directory(ByVal
     Encoder_Index As Long, ByVal Dir As String,)
     As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Save_File_Start(Encoder_Index
     :Longint; n_file_name:String;
     interval_second:Longint;
     format:Longint):Longint;
AngeloMPEG4_Encode_Save_File_Stop(Encoder_Index:
     Longint):Longint;
AngeloMPEG4_Encode_Create_Directory(Encoder_Inde
     x:Longint; Dir:String):Longint;
```

@ Argument

`Encoder_Index`: Indicates the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

`n_file_name`: The argument is the path and name of the file that the encoded image will be saved to.

`interval_second`: This argument is the number of seconds of encoded video to be saved.

Format: The argument describes the format in which to save the file.

1. m4v file
2. avi file
3. both

Dir: The argument is the path and name of the directory that will be created.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int Encoder_Index = 0;
char* n_file_name = "test";
long interval_second = 60;
int format = 3; //save both format
char* Dir = "temp";
Result =
     AngeloMPEG4_Encode_Save_File_Start(Encoder_
     Index, n_file_name, interval_second,
     format);
Result = AngeloMPEG4_Encode_Create_Directory
     (Encoder_Index, Dir);
```

< Visual Basic >

```
Dim Result As Long, Encoder_Index As Long,
     interval_second As Long, format As Long
Encoder_Index = 0;
n_file_name = "test"
interval_second = 60
format = 3 "save both format
Dir = "temp"
Result =
     AngeloMPEG4_Encode_Save_File_Start(Encoder_
     Index, n_file_name, interval_second,
     format)
```

< Delphi >

```
Var
Encoder_Index, Result: Longint;
```

```
Dir, n_file_name: String;
interval_second, format: Longint;
begin
Encoder_Index:= 0;
Dir := "C:\VideoDir";
n_file_name := Dir + "\" + "Video0";
interval_second := 60;
format := 3; // save both format
Result :=
     AngeloMPEG4_Encode_Create_Directory(Encoder
     _Index, Dir);
Result :=
     AngeloMPEG4_Encode_Save_File_Start(Encoder_
     Index, n_file_name, interval_second,
     format);
end;
```

AngeloMPEG4_Encode_Set_Motion_Detection–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Encode_Set_Motion_Detection(int
     Encoder_Index,int Area,int enable, int
     Threshold,int interval,int action,int
     X_Start,int Y_Start,int Width,int Height);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Motion_Detection(ByVal
     Encoder_Index As Long, ByVal Area As Long,
     ByVal enable As Long, ByVal Threshold As
     Long, ByVal interval As Long, ByVal action
     As Long, ByVal X_Start As Long, ByVal
     Y_Start As Long, ByVal Width As Long, ByVal
     Height As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Encode_Set_Motion_Detection(Encoder_
     Index:Longint; Area :Longint;
     enable:Longint; Threshold:Longint;
     interval:Longint; action:Longint;
     X_Start:Longint; Y_Start:Longint;
     Width:Longint; Height:Longint):Longint;
```

@ Argument

Encoder_Index: Indicate the channel index for the MPEG4 encoder. The range of channels is 0 – 15.

Area: User can assign up to 4 motion detection areas in one frame, the valid values are from 1 - 4.

enable:

1: enables motion detection

0: disables motion detection

Threshold: Determines the sensitivity of motion detection measurement. The valid values are from 0 - 15, with 0 being the highest sensitivity.

Interval: The time interval between measurements of motion detection.

Action: This argument describes what actions the function will do.

bit 0: Callback,

X_Start, Y_Start, Width, Height: Sets the boundary of the motion detection area.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int Encoder_Index = 0;
int enable = 1;
int Threshold = 5;
int interval = 3;
int action = 1;
int area =1;
int X_Start = 0;
int Y_Start =0;
int Width = 160;
int Height = 120;
Result =
     AngeloMPEG4_Encode_Set_Motion_Detection(Enc
     oder_Index, area, enable, Threshold,
      interval, action, X_Start, Y_Start, Width,
      Height);
```

< Visual Basic >

```
Dim Result As Long, Encoder_Index As Long, enable
      As Long, Threshold As Long, interval As
      Long, action As Long, area As Long, X_Start
      As Long, Y_Start As Long, Width As Long,
      Height As Long
Encoder_Index = 0
enable = 1
Threshold = 5
interval = 3
action = 1
area =1
X_Start = 0
Y_Start =0
Width = 160
Height = 120
Result =
      AngeloMPEG4_Encode_Set_Motion_Detection(Enc
      oder_Index, area, enable, Threshold,
      interval, action, X_Start, Y_Start, Width,
      Height)
```

<Delphi >

```
var
Encoder_Index, Result: Longint;
enable, Threshold, interval, action: Longint,
      area:Longint, X_Star:Longint,
      Y_Start:Longint, Width: Longint,
      Height:Longint;
begin
Encoder_Index:= 0;
enable := 1;
Threshold := 5;
Interval := 3; // 3 sec
Action := 1; // callback
area =1;
X_Start = 0;
Y_Start =0;
Width = 160;
Height = 120;
if (enable = 1) then
```

```
Result =
      AngeloMPEG4_Encode_Set_Motion_Detection(Enc
      oder_Index, area, enable, Threshold,
      interval, action, X_Start, Y_Start, Width,
      Height)
else // disable motion detection
Result =
      AngeloMPEG4_Encode_Set_Motion_Detection(Enc
      oder_Index, area, 0, Threshold, interval,
      action, X_Start, Y_Start, Width, Height);
end;
```

Decode Functions

@ Name

AngeloMPEG4_Decode_Set_Callback(Decoder_Index, CallBackProc) – Setup the callback function for decoder.

AngeloMPEG4_Decode_Connect(Decoder_Index, Encoder_IP, Encoder_Index) – Connect to the encoder.

AngeloMPEG4_Decode_Disconnect(Decoder_Index) – Disconnect from the encoder.

AngeloMPEG4_Decode_Set_Image_Config(Decoder_Index, ConfigIndex , Value) – Adjust the brightness, contrast, hue etc..

AngeloMPEG4_Decode_Set_Motion_Detection(Decoder_Index, Area, enable, Threshold, interval, action, X_Start, Y_Start, Width, Height) – Set the motion detection criteria, and action when motion occurs in decode site.

AngeloMPEG4_Decode_Get_Config(Decoder_Index, iWidth, iHeight, video_format, color_format, Bitrate, frame_rate) – Get the video property from encode site.

AngeloMPEG4_Decode_Start(Decoder_Index) – Start to decode the video.

AngeloMPEG4_Decode_Stop(Decoder_Index) – Stop decoding the video.

AngeloMPEG4_Decode_Get_FlowRate(Decoder_Index, Byte_Second) – Get the current data flow rate between encoder and decoder

AngeloMPEG4_Decode_ReInitialEx(Decoder_Index, Bitrate, frame_rate, Angelo_Video_Format) – Reset the video property.

AngeloMPEG4_Decode_ReInitial(Decoder_Index, Quality , Angelo_Video_Format) – Reset the video property.

AngeloMPEG4_Decode_Save_File_Start(Decoder_Index, n_file_name, interval_second, format) – Start to save compressed file in decode site.

AngeloMPEG4_Decode_Save_File_Stop(Decoder_Index) – Stop saving compressed file in decode site.

AngeloMPEG4_Decode_File(Decoder_Index, file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame,Time_Seconds) - Decode from *.avi or *.m4v file

AngeloMPEG4_Decode_File_Start(Decoder_Index, Mode) - Start to decode from file

AngeloMPEG4_Decode_File_Set_Position(Decoder_Index, Frame_Index) – Jump to the postion

AngeloMPEG4_Decode_File_Pause(Decoder_Index) - Pauses play

AngeloMPEG4_Decode_File_Continue(Decoder_Index) - Continue the play

AngeloMPEG4_Decode_File_Get_Position(Decoder_Index, Cur_Frame_Index) - Get the current position of play

AngeloMPEG4_AVI_2_M4V(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds) - Translate *avi file into *.m4v file

AngeloMPEG4_M4V_2_AVI(file_name, iWidth, iHeight, Byte_Per_Pixel, Total_Frame, Time_Seconds) - Translate *m4v file into *.avi file

@ Description

`AngeloMPEG4_Decode_Set_Callback:`

This function establishes a notification mechanism between the function library and user process. The callback function is application-defined, users pass the function pointer to function library by calling this function. To receive notification events, users must apply this function before any decode function on the decode site.

`AngeloMPEG4_Decode_ Connect:`

Use this function to establish a connection between decoder and encoder. The video date will then be transferred through this connection.

`AngeloMPEG4_Decode_ Disconnect:`

Use this function to close the connection between decoder and encoder. After closing the connection, the decoder will not receive video data from encoder.

`AngeloMPEG4_Decode_Set_Image_Config:`

If the connection between encoder and decoder is established, use this function to adjust the image property such as contrast and brightness.

`AngeloMPEG4_Decode_Set_Motion_Detection:`

If the connection between encoder and decoder is established, use this function to configure the motion detection criteria and the action when motion occurs in decode site.

`AngeloMPEG4_Decode_Get_Config:`

User must define a callback function, than call "AngeloMPEG4_Decode_Set_Callback". Use "AngeloMPEG4_Decode_ Connect" to establish the connection, if connection is made, the callback function will receive a notification event. The user can then use "AngeloMPEG4_Decode_Get_Config" to retrieve the image configuration such as width, height, bitrate, framerate from the encode site.

`AngeloMPEG4_Decode_Start:`

If the connection between encoder and decoder is established, the video data will transfer from encoder to decoder automatically. Use this function to restart the video data transmission, if “AngeloMPEG4_Decode_Stop” has been called to stop the transmission.

`AngeloMPEG4_Decode_Stop:`

This function only stops the video data transmission between decoder and encoder, but the connection is still established.

`AngeloMPEG4_Decode_Get_FlowRate:`

If the connection between encoder and decoder is established, use this function to query the current data flow rate between encode and decode.

`AngeloMPEG4_Decode_ReInitialEx:`

Because the Bitrate, frame_rate is initialized in the encode site, the decode uses this function to reset the image quality if connection is established.

Note: If one decoder changes the quality, the others will also have a different image quality.

`AngeloMPEG4_Decode_ReInitial:`

Because the Bitrate, frame_rate is initialized in encode site, the decode use this function to reset the image quality, if the connection is established.

`AngeloMPEG4_Decode_Save_File_Start:`

If the connection between encoder and decoder is established, use this function to save the encoded image into an “.avi”, “.m4v” video file on the decode site. The .avi file is the standard video format, and .m4v is only accessible in this function library.

Note:

1. Do not add the file extension name.
2. Users must install the XVID Codec from the setup disk. The “.avi” file can be played in MS Media Player.

`AngeloMPEG4_Decode_Save_File_Stop:`

If the connection between encoder and decoder is established, use this function to stop saving video file on the decode site. In general, the video file will close automatically after the “Interval” parameter in AngeloMPEG4_Decode_Save_File_Start.

`AngeloMPEG4_Decode_File:`

If you save the video file into “.m4v” or “.avi”, and the file is closed, than you can use this function to decode the “.m4v” or “.avi”, and get the video image in callback function, than you can draw the image on the Windows DC. This function initialize the decode from file

`AngeloMPEG4_Decode_File_Start:`

Start decoding from file. If the callback function has been set up, a video buffer of each frame will be received.

`AngeloMPEG4_Decode_File_Set_Position:`

Skip some frames, and jump to the frame you want. You can get the total frames of the file using AngeloMPEG4_Decode_File.

`AngeloMPEG4_Decode_File_Pause:`

The file is paused until AngeloMPEG4_Decode_File_Continue is activated.

`AngeloMPEG4_Decode_File_Get_Position:`

Get the current frame index of the file.

`AngeloMPEG4_AVI_2_M4V:`

Use this function to translate a closed “.avi” video file into “.m4v” format.

`AngeloMPEG4_ M4V_2_AVI:`

Use this function to translate a closed “.m4v” video file into “.avi” format.

AngeloMPEG4_Decode_Connect –

AngeloMPEG4_Decode_Disconnect –

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Connect(int Decoder_Index,
char* Encoder_IP, unsigned int Enocder_Index);
int AngeloMPEG4_Decode_Disconnect(int
Decoder_Index);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Connect(ByVal Decoder_Index As
Long, ByVal Encoder_IP As String, ByVal
Enocder_Index As Long) As Long
AngeloMPEG4_Decode_ Disconnect (ByVal
Decoder_Index As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Connect(Decoder_Index:Longint
; Encoder_IP:String;
Encoder_Index:Longint):Longint;
AngeloMPEG4_Decode_Disconnect(Decoder_Index:Long
int):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

`Encoder_IP`: The IP address of MPEG4 Encode.

`Encoder_Index`: The channel of MPEG4 Encoder.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
char* Encoder_IP = "127.0.0.1"; //localhost
unsigned int Encoder_channel = 0;
```

```
Result = AngeloMPEG4_Decode_Connect(channel,
     Encoder_IP, Encoder_channel);
Result = AngeloMPEG4_Decode_Disconnect(channel);
```

< Visual Basic >

```
Dim Result As Long, channel As Long,
     Encoder_channel As Long
Dim Encoder_IP As String
channel = 0
Encoder_IP = "127.0.0.1" 'localhost
Encoder_channel = 0
Result = AngeloMPEG4_Decode_Connect(channel,
     Encoder_IP, Encoder_channel)
Result = AngeloMPEG4_Decode_Disconnect(channel)
```

<Delphi >

```
var
channel: Longint;
Encoder_IP: String;
Encoder_channel: Longint;
Result: Longint;
begin
channel := 0;
Remote_IP := '127.0.0.1'; //localhost
Result := AngeloMPEG4_Decode_Connect(channel,
     Encoder_IP, Encoder_channel);
Result := AngeloMPEG4_DecodeDisconnect(channel);
end;
```

AngeloMPEG4_Decode_Set_Callback–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Set_Callback(int
Decoder_Index, void ( __stdcall
*CallBackProc)(int channel, long int_status, long
VideoBufferaddress));
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Callback(ByVal
Decoder_Index As Long, ByVal CallBack As Long) As
Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Callback(Decoder_Index:Lo
ngint;
lpDecodeCallBackProc:DecodeCallBackProc):Longint
;
```

@ Argument

`Decoder_Index`: Indicates the channel number of Decoder. The range of channel is 0 - 15.

`int_status`:

`Interrupt status`:

Bit 0: Image ready

Bit 1: Motion Detection occur

Bit 2: Connection establish

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
void __stdcall Decode_Callback(int channel, long
     int_status, long VideoBufferaddress)
{
     if((int_status & 0x01) == 1) //Image Ready
     {
          //Start Drawing

     memcpy(Temp,(PVOID)VideoBufferaddress,iWidt
     h*iHeight*3);
          gpDC-
     >BitBlt(10,10,iWidth,iHeight,MemDC,0,0,SRCC
     OPY);
     }
     if((int_status>>1 & 0x01) == 1) //
     MotionDetection Occur
     {
          //Deal with MotionDetection
          Beep(1024, 100);
     }
     if((int_status>>2 & 0x01) == 1) //Connection
     establish
     {
          //Prepare DC for Preview
          int Bitrate = 0, frame_rate = 0,
     colorspace = 0;

          AngeloMPEG4_Decode_Get_Config(channel,
     &iWidth, &iHeight, &videoformat,
     &colorspace, &Bitrate, &frame_rate);
          }
}
Result = AngeloMPEG4_Decode_Set_Callback(channel,
     Decode_Callback);
```

< Visual Basic >

```
Dim Result As Long, channel As Long
     Public Sub lpcallback(ByVal channel As Long,
     ByVal int_status As Long, ByVal
     VideoBufferaddress As Long)
               If int_status And &H2 Then
     'detected motion
               ElseIf int_status And &H4 Then '
     connect to encoder
               ElseIf int_status And &H1 Then '
     image ready
               End If
     End Sub
     Result =
     AngeloMPEG4_Decode_Set_Callback(channel,
     AddressOf lpcallback)
```

< Delphi >

```
procedure DecoderCallbackProc(channel:Longint;
     int_status:Longint;
     VideoBufferaddress:Longint); stdcall
var
    Str_Addr: Pointer;
    Bitrate, Framerate, colorspace, videoformat:
     Longint;
begin
    case int_status of
```

```
            1: begin  {********* image buffer OK
           *********}
              // draw image here
            end;
            2: begin  {********* Motion Detected
           *********}
            end;
           4: begin  {********* Connect Ready Interrupt
           *********}
              // You can get image config here and do
           somthing
            end;
         end;  // end case int_status of
     end;
     // Main Code
     var
     channel: Longint;
     Result: Longint;
     begin
     channel := 0;
     Result :=
           AngeloMPEG4_Decode_Set_Callback(channel,
           DecoderCallbackProc);
     end;
```

AngeloMPEG4_Decode_Set_Image_Config–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Set_Image_Config(int
     Decoder_Index, int ConfigIndex , int Value);
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Image_Config(ByVal
     channel As Long, ByVal Decoder_Index As
     Long, ByVal Value As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Image_Config(Decoder_Inde
     x:Longint; ConfigIndex:Longint;
     Value:Longint):Longint;
```

@ Argument

`Decoder_Index:` Indicate the channel number of Decoder. The range of channel is 0 ~ 15.

`ConfigIndex:`

0 for BRIGHTNESS

1 for HUE

2 for SATURATION (U)

3 for SATURATION (V)

4 for CONTRAST (LUMA)

5 for luma notch filter (for monochrome video, the notch filter should not be used)

value: (0-255):

	Range	Default value
BRIGHTNESS	0 - 255	128
HUE	0 - 255	0
CHROMA (U)	0 - 255	127
CHROMA (V)	0 - 255	127
LUMA	0 - 255	108
LUMA notch filter	0 (Enable) or 1 (Disable)	

Table 6-4: Video adjustments table

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int ConfigIndex = 0;
int value = 128;
Result =
     AngeloMPEG4_Decode_Set_Image_Config(channel
     , ConfigIndex, value);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, ConfigIndex
     As Long, value As Long
channel = 0
ConfigIndex = 0
value = 128
Result =
     AngeloMPEG4_Decode_Set_Image_Config(channel
     , ConfigIndex, value)
```

<Delphi >

```
var
channel: Longint;
ConfigIndex: Longint;
Value: Longint;
Result: Longint;
begin
channel := 0;
ConfigIndex := 0;
Value := 128;
Result :=
     AngeloMPEG4_Decode_Set_Image_Config(channel
     , ConfigIndex, Value);
end;
```

AngeloMPEG4_Decode_Set_Motion_Detection–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Set_Motion_Detection(int
     Decoder_Index,int Area,int enable, int
     Threshold,int interval,int action,int
     X_Start,int Y_Start,int Width,int Height);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Motion_Detection(ByVal
     Decoder_Index As Long, ByVal enable As Long,
     ByVal Threshold As Long, ByVal interval As
     Long, ByVal action As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Set_Motion_Detection(Decoder_
     Index:Longint; Area :Longint;
     enable:Longint; Threshold:Longint;
```

```
        interval:Longint; action:Longint;
        X_Start:Longint; Y_Start:Longint;
        Width:Longint; Height:Longint):Longint;
```

@ Argument

`Decoder_Index:` Indicates the channel number of Decoder. The range of channel is 0 - 15.

`Area:` User can assign up to four motion detection areas in one frame, the valid value range is 1 - 4.

`Enable:`

1: enable Motion Detection

0: disable Motion Detection

`Threshold:` The threshold senses motion detection occurrence. The value range is 0 - 15, with 0 being the highest sensitivity.

`Interval:` Time interval measures motion detection occurrence.

`Action:` The argument descript what actions the function will do.

bit 0: Callback,

bit 1: Reserved,

bit 2: Send motion frame

Example: when action = 1 + 4, the function will perform callback and send the motion image.

`X_Start, Y_Start, Width, Height:` Set the boundary of motion detection area.

@ Return Code

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int enable = 1;
int Threshold = 5;
int interval = 3;
int action = 1 + 4;
int area =1;
int X_Start = 0;
int Y_Start =0;
int Width = 160;
int Height = 120;
Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, enable, Threshold, interval,
     action, X_Start, Y_Start, Width, Height);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, enable As
     Long, Threshold As Long, interval As Long,
     action As Long, area As Long, X_Start As
     Long, Y_Start As Long, Width As Long, Height
     As Long
channel = 0
enable = 1
Threshold = 5
interval = 3
action = 1 + 4
area =1
X_Start = 0
Y_Start =0
Width = 160
Height = 120

Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, enable, Threshold, interval,
     action, X_Start, Y_Start, Width, Height)
```

< Delphi >

```
var
channel, Result: Longint;
enable, Threshold, interval, action: Longint,
     area:Longint, X_Star:Longint,
     Y_Start:Longint, Width: Longint,
     Height:Longint;
begin
channel := 0;
enable := 1;
Threshold := 5;
```

```
Interval := 3; // 3 sec
Action := 1+4; // callback & send motion image
area =1;
X_Start = 0;
Y_Start =0;
Width = 160;
Height = 120;
if (enable = 1) then
Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, enable, Threshold, interval,
     action, X_Start, Y_Start, Width, Height)
else // disable motion detection
Result =
     AngeloMPEG4_Decode_Set_Motion_Detection(cha
     nnel, area, 0, Threshold, interval, action,
     X_Start, Y_Start, Width, Height);
end;
```

AngeloMPEG4_Decode_Get_Config–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Get_Config(int
     Decoder_Index, int* iWidth, int* iHeight,
     int* video_format, int* color_format, int*
     Bitrate, int* frame_rate);
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_Config (ByVal
     Decoder_Index As Long, ByRef iWidth As Long,
     ByRef iHeight As Long, ByRef video_format As
     Long, ByRef color_format As Long, ByRef
     Bitrate As Long, ByRef frame_rate As Long)
     As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_Config(Decoder_Index:Long
     int; var iWidth:Longint; var
     iHeight:Longint; var video_format:Longint;
     var color_format:Longint; var
     Bitrate:Longint; var
     frame_rate:Longint):Longint;
```

@ Argument

`Decoder_Index:` Indicates the channel number of Decoder. The range of channel is 0 - 15.

`iWidth:` Indicates the width of the MPEG4 image size.

`iHeight:` Indicates the height of the MPEG4 image size.

`video_format:`

Full NTSC (640*480)	= 0,
Full PAL (768*576)	= 1,
CIF NTSC (320*240)	= 2,
CIF PAL (384*288)	= 3,
QCIF NTSC (160*120)	= 4,
QCIF PAL (192*144)	= 5,

`color_format:`

RGB16	= 0,
GRAY	= 1,
RGB15	= 2,
RGB24	= 3,
RGB32	= 4,
RGB8	= 5,
RAW8X	= 6,
YUY24:2:2	= 7,
BtYUV 4:1:1	= 8

At present, we only provide RGB24 color format, hence the value should always be set at 3.

`Bitrate:` Indicates the bitrate of MPEG4 stream from the encode server.

`frame_rate:` Indicates the frame rate of MPEG4 stream from the encode server.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int iWidth = 0;
int iHeight = 0;
int video_format = 0;
int color_format = 0;
int Bitrate = 0;
int frame_rate = 0;
Result = AngeloMPEG4_Decode_Get_Config(channel,
      &iWidth, &iHeight, &videoformat,
      &color_format, &Bitrate, &frame_rate);
```

< Visual Basic >

```
      Dim Result As Long, channel As Long, iWidth
      As Long, iHeight As Long, video_format As
      Long, color_format As Long, Bitrate As Long,
      frame_rate As Long
Channel = 0
Result = AngeloMPEG4_Decode_Get_Config(channel,
      iWidth, iHeight, videoformat, colorformat,
      Bitrate, frame_rate)
```

< Delphi >

```
var
channel: Longint;
iWidth, iHeight: Longint;
videoformat, colorspace, Bitrate, frame_rate:
      Longint;
Result: Longint;
begin
channel := 0;
Result := AngeloMPEG4_Decode_Get_Config(channel,
      iWidth, iHeight, videoformat, colorspace,
      Bitrate, frame_rate);
end;
```

AngeloMPEG4_Decode_Start–

AngeloMPEG4_Decode_Stop–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Start(int Decoder_Index);
int AngeloMPEG4_Decode_Stop(int Decoder_Index);
```

Visual Basic(Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Start(ByVal Decoder_Index As
     Long) As Long
AngeloMPEG4_Decode_Stop(ByVal Decoder_Index As
     Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Start(Decoder_Index:Longint):
     Longint;
AngeloMPEG4_Decode_Stop(Decoder_Index:Longint):L
     ongint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of Decoder. The range of channel is 0 - 15.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
Result = AngeloMPEG4_Decode_Start(channel);
Result = AngeloMPEG4_Decode_Stop(channel);
```

< Visual Basic >

```
Dim Result As Long, channel As Long
channel = 0
Result = AngeloMPEG4_Decode_Start(channel)
Result = AngeloMPEG4_Decode_Stop(channel)
```

< Delphi >

```
var
channel: Longint;
Result: Longint;
begin
channel := 0;
Result := AngeloMPEG4_Decode_Start(channel);
Result := AngeloMPEG4_Decode_Stop(channel);
end;
```

AngeloMPEG4_Decode_Get_FlowRate–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_Get_FlowRate(int
     Decoder_Index, long* Byte_Second);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_FlowRate(ByVal
     Decoder_Index As Long, ByRef flow_rate As
     Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Get_FlowRate(Decoder_Index:Lo
     ngint; var Byte_Second:Longint):Longint;
```

@ Argument

Decoder_Index: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

Byte_Second: The current flow rate of MPEG4 streaming measured in Byte/sec.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
long Byte_Second;
Result = AngeloMPEG4_Decode_Get_FlowRate(channel,
     &Byte_Second);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, Byte_Second
      As Long
Result = AngeloMPEG4_Decode_Get_FlowRate(channel,
      Byte_Second)
```

<Delphi >

```
AngeloMPEG4_Decode_Get_FlowRate –
var
channel: Longint;
Byte_Second: Longint;
Result: Longint;
begin
channel := 0;
Result :=
      AngeloMPEG4_Decode_Get_FlowRate(channel,
      Byte_Second);
end;
```

AngeloMPEG4_Decode_ ReInitial–

AngeloMPEG4_Decode_ ReInitialEx–

@ Syntax

C/C++ (Windows 98/NT/2000/XP)

```
int AngeloMPEG4_Decode_ReInitial(int
      Decoder_Index, int Quality, int
      Angelo_Video_Format);
int AngeloMPEG4_Decode_ReInitialEx(int
      Decoder_Index, int Bitrate, int frame_rate,
      int Angelo_Video_Format);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_ReInitial (ByVal Decoder_Index
      As Long, ByVal Quality As Long, ByVal
      Video_Format As Long) As Long
AngeloMPEG4_Decode_ReInitialEx (ByVal
      Decoder_Index As Long, ByVal Bitrate As
      Long, ByVal frame_rate As Long, ByVal
      Video_Format As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_ReInitial(Decoder_Index:Longi
     nt; Quality:Longint;
     Angelo_Video_Format:Longint):Longint;
AngeloMPEG4_Decode_ReInitialEx(Decoder_Index:Lon
     gint; Bitrate:Longint; frame_rate:Longint;
     Angelo_Video_Format:Longint):Longint;
```

@ Argument

Decoder_Index: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

Quality:

Quality	value	image 640*480	image 320*240	image 160*120

Lowest	-2	Bitrate = 320000*4
		frame_rate = 3
			Bitrate = 320000
			frame_rate = 3
				Bitrate = 240000/4
				frame_rate = 3

Low	-1	Bitrate = 400000*2
		frame_rate = 6
			Bitrate = 400000
			frame_rate = 6
				Bitrate = 400000/4
				frame_rate = 6

Normal	0	Bitrate = 480000*4
		frame_rate = 15
			Bitrate = 480000
			frame_rate = 15
				Bitrate = 480000/4
				frame_rate = 15

High	1	Bitrate = 512000*4
		frame_rate = 30
			Bitrate = 512000
			frame_rate = 30
				Bitrate = 512000/4
				frame_rate = 30

Highest	2	Bitrate = 1024000*4
		frame_rate = 30
			Bitrate = 1024000
			frame_rate = 30
				Bitrate = 1024000/4
				frame_rate = 30

Table 6-5: Video quality table

Bitrate: Indicates the bitrate of MPEG4 stream from encode server.

Frame_rate: Indicates the frame rate of MPEG4 stream from encode server. The values range is 0 - 30.

Angelo_Video_Format:

Full NTSC (640*480) = 0,

Full PAL (768*576) = 1,

CIF NTSC (320*240) = 2,

CIF PAL (384*288) = 3,

QCIF NTSC (160*120) = 4,

QCIF PAL (192*144) = 5,

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
int Quality =0
int Bitrate = 480000;
int frame_rate = 15;
int Angelo_Video_Format = 2;
Result = AngeloMPEG4_Decode_ReInitia(channel,
     Quality, Angelo_Video_Format);
Result = AngeloMPEG4_Decode_ReInitialEx(channel,
     Bitrate, frame_rate, Angelo_Video_Format);
```

< Visual Basic >

```
Dim Result As Long, channel As Long, Quality As
     Long, Bitrate As Long, frame_rate As Long,
     Angelo_Video_Format As Long
channel = 0
Quality =0
Bitrate = 480000
frame_rate = 15
Angelo_Video_Format = 2
Result = AngeloMPEG4_Decode_ReInitia(channel,
     Quality, Angelo_Video_Format)
Result = AngeloMPEG4_Decode_ReInitiaEx(channel,
     Bitrate, frame_rate, Angelo_Video_Format)
```

< Delphi >

```
var
channel: Longint;
Quality, Bitrate, frame_rate,
     Angelo_Video_Format: Longint;
Result: Longint;
begin
channel := 0;
Quality :=0;
Bitrate := 480000;
frame_rate := 15;
```

```
Angelo_Video_Format = 2;
Result = AngeloMPEG4_Decode_ReInitial(channel,
      Quality, Angelo_Video_Format);
Result = AngeloMPEG4_Decode_ReInitialEx(channel,
      Bitrate, frame_rate, Angelo_Video_Format);
end;
```

AngeloMPEG4_Decode_ Save_File_Start–

AngeloMPEG4_Decode_ Save_File_Stop–

@ Syntax

C/C++ (Windows 98/NT/2000/XP/CE.NET)

```
int AngeloMPEG4_Decode_Save_File_Start(int
      Decoder_Index, char* n_file_name, long
      interval_second, long format);
int AngeloMPEG4_Decode_Save_File_Stop(int
      Decoder_Index);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Save_File_Start (ByVal
      Decoder_Index As Long, ByVal n_file_name As
      String, ByVal interval_second As Long, ByVal
      format As Long) As Long
AngeloMPEG4_Decode_Save_File_Stop (ByVal
      Decoder_Index As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_Save_File_Start(Decoder_Index
      :Longint; n_file_name:String;
      interval_second:Longint;
      format:Longint):Longint;
AngeloMPEG4_Decode_Save_File_Stop(Decoder_Index:
      Longint):Longint;
```

@ Argument

`Decoder_Index`: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

`n_file_name`: The name to save the file to, excludes the extension of file name.

`interval_second`: Specify the save time for MPEG4 streaming.

format:

1: m4v,

2: avi.

3: both.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int channel = 0;
char* n_file_name = “test”;
int interval_second = 10; //10 seconds
long format = 1 + 2; //save both file format
Result =
     AngeloMPEG4_Decode_Save_File_Start(channel,
     n_file_name, interval_second, format);
```

< Visual Basic >

```
Dim Result As Long, channel As Long,
     interval_second As Long, format As Long

channel = 0;
n_file_name = “test”
interval_second = 60
format = 3 ‘save both format
Result =
     AngeloMPEG4_Decode_Save_File_Start(channel,
     n_file_name, interval_second, format)
```

<Delphi >

```
var
channel: Longint;
n_file_name: String;
interval_second, format: Longint;
Result: Longint;
begin
channel := 0;
n_file_name := ‘Video0’;
interval_second := 10;
```

```
format := 3; // Save both format
Result :=
     AngeloMPEG4_Decode_Save_File_Start(channel,
     n_file_name, interval_second, format);
end;
```

AngeloMPEG4_Decode_File–

AngeloMPEG4_Decode_File_Start–

AngeloMPEG4_Decode_File_Set_Position–

AngeloMPEG4_Decode_File_Pause–

AngeloMPEG4_Decode_File_Continue–

AngeloMPEG4_Decode_File_Get_Position–

AngeloMPEG4_AVI_2_M4V–

AngeloMPEG4_M4V_2_AVI–

@ Syntax

C/C++ (Windows 98/NT/2000/XP/CE.NET)

```
int AngeloMPEG4_Decode_File(int
     Decoder_Index,char* file_name,unsigned
     long* iWidth,unsigned long*
     iHeight,unsigned long*
     Byte_Per_Pixel,unsigned long*
     Total_Frame,unsigned long* Time_Seconds);
int AngeloMPEG4_Decode_File_Start(int
     Decoder_Index,int Mode);
int AngeloMPEG4_Decode_File_Set_Position(int
     Decoder_Index, long* Frame_Index);
int AngeloMPEG4_Decode_File_Pause(int
     Decoder_Index);
int AngeloMPEG4_Decode_File_Continue(int
     Decoder_Index);
int AngeloMPEG4_Decode_File_Get_Position(int
     Decoder_Index, long* Cur_Frame_Index);
```

```
int AngeloMPEG4_AVI_2_M4V(char*
     file_name,unsigned long* iWidth,unsigned
     long* iHeight,unsigned long*
     Byte_Per_Pixel,unsigned long*
     Total_Frame,unsigned long* Time_Seconds);
int AngeloMPEG4_M4V_2_AVI(char*
     file_name,unsigned long* iWidth,unsigned
     long* iHeight,unsigned long*
     Byte_Per_Pixel,unsigned long*
     Total_Frame,unsigned long* Time_Seconds);
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_File (ByVal Decoder_Index As
     Long, ByVal file_name As String, iWidth As
     Long, iHeight As Long, Byte_Per_Pixel As
     Long, Total_Frame As Long, Time_Seconds As
     Long) As Long
AngeloMPEG4_Decode_File_Start (ByVal
     Decoder_Index As Long, ByVal Mode As Long)
     As Long
AngeloMPEG4_Decode_File_Set_Position (ByVal
     Decoder_Index As Long, Frame_Index As Long)
     As Long
AngeloMPEG4_Decode_File_Pause (ByVal
     Decoder_Index As Long) As Long
AngeloMPEG4_Decode_File_Continue (ByVal
     Decoder_Index As Long) As Long
AngeloMPEG4_Decode_File_Get_Position (ByVal
     Decoder_Index As Long, Cur_Frame_Index As
     Long) As Long
AngeloMPEG4_AVI_2_M4V (ByVal file_name As String,
     iWidth As Long, iHeight As Long,
     Byte_Per_Pixel As Long, Total_Frame As Long,
     Time_Seconds As Long) As Long
AngeloMPEG4_M4V _2_ AVI (ByVal file_name As
     String, iWidth As Long, iHeight As Long,
     Byte_Per_Pixel As Long, Total_Frame As Long,
     Time_Seconds As Long) As Long
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Decode_File(Decoder_Index:Longint;
     file_name:String; var iWidth:Longint; var
     iHeight:Longint; var
```

```
          Byte_Per_Pixel:Longint; var
          Total_Frame:Longint; var
          Time_Seconds:Longint):Longint;
     AngeloMPEG4_Decode_File_Start(Decoder_Index:Long
          int; Mode:Longint):Longint;
     AngeloMPEG4_Decode_File_Set_Position(Decoder_Ind
          ex:Longint; var
          Frame_Index:Longint):Longint;
     AngeloMPEG4_Decode_File_Pause(Decoder_Index:Long
          int):Longint;
     AngeloMPEG4_Decode_File_Continue(Decoder_Index:L
          ongint):Longint;
     AngeloMPEG4_Decode_File_Get_Position(Decoder_Ind
          ex:Longint; var
          Cur_Frame_Index:Longint):Longint;
     AngeloMPEG4_AVI_2_M4V(file_name:String; var
          iWidth:Longint; var iHeight:Longint; var
          Byte_Per_Pixel:Longint; var
          Total_Frame:Longint; var
          Time_Seconds:Longint):Longint;
     AngeloMPEG4_M4V_2_AVI(file_name:String; var
          iWidth:Longint; var iHeight:Longint; var
          Byte_Per_Pixel:Longint; var
          Total_Frame:Longint; var
          Time_Seconds:Longint):Longint;
```

@ Argument

Decoder_Index: Indicates the channel number of MPEG4 Decoder. The range of channel is 0 - 15.

file_name: The name of file to save to, includes the path and extension of file name.

iWidth: Indicate the width of the MPEG4 image size.

iHeight: Indicates the height of the MPEG4 image size.

Byte_Per_Pixel: Number of Bytes per Pixel

Total_Frame: Number of frames in the MPEG4 file.

Time_Seconds: The total time of the MPEG4 file in seconds.

Mode: The play mode of the Mpeg4 file

0: Play once

1: Repeat

Frame_Index: Zero index of the frame

Cur_Frame_Index: Current frame index

PlayFactor: The speed to play the MPEG4 file

1: Normal

2: 2x faster

-2: 2x slower

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
int m_Decoder_Channel = 0;
long Width=0;
long Height=0;
long Byte_Pixel=0;
long m_total_frame=0;
long m_Time_Seconds=0;
long m_pos=0;
long Mode = 0; //play once
char* m_filename = "test1.m4v";
char* m4v_filename = "test2.m4v";
char* avi_filename = "test3.avi";
void CM4VPlayerView::MediaStreamProc( int
     Decoder_Channel ,long int_status,long
     VideoBufferaddress )
{
     …
     …
}
AngeloMPEG4_Decode_Set_Callback(m_Decoder_Channe
     l,MediaStreamProc);
AngeloMPEG4_Decode_File(m_Decoder_Channel,m_file
     name,&Width,&Height,&Byte_Pixel,&m_total_fr
     ame,&m_Time_Seconds);
AngeloMPEG4_Decode_File_Start(m_Decoder_Channel,
     Mode);
AngeloMPEG4_Decode_File_Set_Position(m_Decoder_C
     hannel,& m_total_frame/2);
```

```
AngeloMPEG4_Decode_File_Pause(m_Decoder_Channel)
     ;
AngeloMPEG4_Decode_File_Continue(m_Decoder_Chann
     el);
AngeloMPEG4_Decode_File_Get_Position(m_Decoder_C
     hannel,&m_pos);
AngeloMPEG4_Decode_Stop(m_Decoder_Channel);
AngeloMPEG4_Decode_M4V_2_AVI(m4v_filename,&Width
     ,&Height,&Byte_Pixel,&m_total_frame,&m_Time
     _Seconds);
AngeloMPEG4_Decode_AVI_2_M4V(avi_filename,&Width
     ,&Height,&Byte_Pixel,&m_total_frame,&m_Time
     _Seconds);
```

< Visual Basic >

```
Dim Result As Long, m_Decoder_Channel As Long,
     Width As Long, Height As Long, Byte_Pixel As
     Long, m_total_frame As Long, m_Time_Seconds
     As Long, m_pos As Long
Dim m_filename As String, m4v_filename As String,
     avi_filename As String,

m_filename = “test1.m4v”
m4v_filename = “test2.m4v”
avi_filename = “test3.avi”
m_Decoder_Channel = 0
Mode = 0 ‘play once

Public Sub lpcallback(ByVal Decoder_Index As
     Long, ByVal int_status As Long, ByVal
     VideoBufferaddress As Long)
…
…
End Sub
Result =
     AngeloMPEG4_Decode_Set_Callback(m_Decoder_C
     hannel, AddressOf lpcallback)
Result=AngeloMPEG4_Decode_File(m_Decoder_Channel
     ,m_filename,Width,Height,Byte_Pixel,m_total
     _frame,m_Time_Seconds)
Result =
     AngeloMPEG4_Decode_File_Start(m_Decoder_Cha
     nnel,Mode)
```

```
Result =
      AngeloMPEG4_Decode_File_Set_Position(m_Deco
      der_Channel,m_total_frame/2)
Result =
      AngeloMPEG4_Decode_File_Pause(m_Decoder_Cha
      nnel)
Result =
      AngeloMPEG4_Decode_File_Continue(m_Decoder_
      Channel)
Result =
      AngeloMPEG4_Decode_File_Get_Position(m_Deco
      der_Channel,m_pos)
Result =
      AngeloMPEG4_Decode_Stop(m_Decoder_Channel)
Result=AngeloMPEG4_Decode_M4V_2_AVI(m4v_filename
      ,Width,Height,Byte_Pixel,m_total_frame,m_Ti
      me_Seconds)
Result=AngeloMPEG4_Decode_AVI_2_M4V(avi_filename
      ,Width,Height,Byte_Pixel,m_total_frame,m_Ti
      me_Seconds)
```

< Delphi >

```
procedure
      DecoderCallbackProc(Decoder_Index:Longint;i
      nt_status:Longint;VideoBufferaddress:Longin
      t); stdcall
var
     Str_Addr : Pointer;
     Bitrate, Framerate, colorspace,videoformat :
      Longint;
begin
          …
          …

end;
…
…


var
m_filename, m4v_filename, avi_filename: String;
Result, m_Decoder_Channel, Width, Height,
      Byte_Pixel, m_total_frame, m_Time_Seconds,
      m_pos, Mode: Longint;
```

```
begin
m_Decoder_Channel:= 0;
Mode := 0; //play once
m_filenam := 'test1.m4v';
m4v_filename:= 'test2.m4v';
avi_filename := 'test3.avi';

Result :=
     AngeloMPEG4_Decode_Set_Callback(m_Decoder_C
     hannel, DecoderCallbackProc);
Result
     :=AngeloMPEG4_Decode_File(m_Decoder_Channel
     ,m_filename,Width,Height,Byte_Pixel,m_total
     _frame,m_Time_Seconds);
Result :=
     AngeloMPEG4_Decode_File_Start(m_Decoder_Cha
     nnel,Mode);
Result :=
     AngeloMPEG4_Decode_File_Set_Position(m_Deco
     der_Channel,m_total_frame div 2);
Result :=
     AngeloMPEG4_Decode_File_Pause(m_Decoder_Cha
     nnel);
Result :=
     AngeloMPEG4_Decode_File_Continue(m_Decoder_
     Channel);
Result :=
     AngeloMPEG4_Decode_File_Get_Position(m_Deco
     der_Channel,m_pos);
Result :=
     AngeloMPEG4_Decode_Stop(m_Decoder_Channel);
Result:=AngeloMPEG4_Decode_M4V_2_AVI(m4v_filenam
     e,Width,Height,Byte_Pixel,m_total_frame,m_T
     ime_Seconds);
Result:=AngeloMPEG4_Decode_AVI_2_M4V(avi_filenam
     e,Width,Height,Byte_Pixel,m_total_frame,m_T
     ime_Seconds);
end;
```

System Functions

@ Name

AngeloMPEG4_Get_Version(lMpeg4_DLLVersion, AngeloRTV_DLLVersion, Reserved)

@ Description

`AngeloMPEG4_Get_Version:` Use this function to get the software information.

AngeloMPEG4_Get_Version –

@ Syntax

C/C++ (Windows 98/NT/2000/XP/CE.NET)

```
int AngeloMPEG4_Get_Version(long
      *Mpeg4_DLLVersion, long
      *AngeloRTV_DLLVersion, long *Reserved);
```

Delphi (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Get_Version(var
      Mpeg4_DLLVersion:Longint; var
      AngeloRTV_DLLVersion:Longint; var
      Reserved:Longint):Longint;
```

Visual Basic (Windows 98/NT/2000/XP)

```
AngeloMPEG4_Get_Version(ByRef
      AngeloMpeg4_DLLVersion As Long, ByRef
      AngeloRTV_DLLVersion As Long, ByRef Reserved
      As Long) As Long
```

@ Argument

`Mpeg4_DLLVersion:` Indicates the current version of the MPEG4 DLL. It is of 4 rows in length.

`AngeloRTV_DLLVersion:` Indicates the current version of AngeloRTV DLL. It is of 4 rows in length.

@ Return Code

0: ERROR_NoError

@ Example

< VC/BCB >

```
int Result;
```

```
long Mp4Version[4] = {0}, DLLVersion[4] = {0},
      VersionReserved[4] = {0};
CString str1, str2;
Result = AngeloMPEG4_Get_Version(Mp4Version,
      DLLVersion, VersionReserved);
str1.Format("%d.%d.%d.%d", DLLVersion[0],
      DLLVersion[1], DLLVersion[2],
      DLLVersion[3]);
str2.Format("%d.%d.%d.%d", Mp4Version[0],
      Mp4Version[1], Mp4Version[2],
      Mp4Version[3]);
```

< Visual Basic >

```
Dim Result As long, Mp4Version(0 to 3) As Long,
      DLLVersion(0 to 3) As Long,
      VersionReserved(0 to 3) As Long
Result = AngeloMPEG4_Get_Version(Mp4Version(0),
      DLLVersion(0), VersionReserved(0))
```

< Delphi >

```
var
Mpeg4_DLLVersion : array[0..3] of Longint;
AngeloRTV_DLLVersion : array[0..3] of Longint;
Reserved : array[0..3] of Longint;
Result: Longint;
Str_AngeloMPEG4_Version, Str_AngeloRTV_Version:
      String;
begin
Result :=
      AngeloMPEG4_Get_Version(Mpeg4_DLLVersion[0]
      , AngeloRTV_DLLVersion[0], Reserved[0]);
Str_AngeloMPEG4_Version :=
      IntToStr(Mpeg4_DLLVersion[0]);
Str_AngeloMPEG4_Version :=
      Str_AngeloMPEG4_Version + “.” +
      IntToStr(Mpeg4_DLLVersion[1]);
Str_AngeloMPEG4_Version :=
      Str_AngeloMPEG4_Version + “.” +
      IntToStr(Mpeg4_DLLVersion[2]);
Str_AngeloMPEG4_Version :=
      Str_AngeloMPEG4_Version + “.” +
      IntToStr(Mpeg4_DLLVersion[3]);
```

```
Str_AngeloRTV_Version :=
      IntToStr(AngeloRTV_DLLVersion[0]);
Str_AngeloRTV_Version := Str_AngeloRTV_Version +
      "." + IntToStr(AngeloRTV_DLLVersion[1]);
Str_AngeloRTV_Version := Str_AngeloRTV_Version +
      "." + IntToStr(AngeloRTV_DLLVersion[2]);
Str_AngeloRTV_Version := Str_AngeloRTV_Version +
      "." + IntToStr(AngeloRTV_DLLVersion[3]);
end;
```

6.5 Hardware reference

Please refer to the GEME-V3000 Series User's Manual for detailed information regarding hardware.

Safety Instructions

Please read and follow all instructions marked on the product and in the documentation before operating the system. Retain all safety and operating instructions for future use.

- Please read these safety instructions carefully.
- Please keep this User's Manual for future reference.
- The equipment should be operated in an ambient temperature between -10 to 55.5°C.
- The equipment should be operated only from the type of power source indicated on the rating label. Make sure the voltage of the power source is correct when connecting the equipment to the power outlet.
- If your equipment has a voltage selector switch, make sure the switch is set to the proper position for your area. The voltage selector switch is set at the factory to the correct voltage.
- For pluggable equipment, ensure that an electrical outlet is installed nearby and is easily accessible.
- Secure the power cord to prevent unnecessary accidents. Do not place anything over the power cord.
- If the equipment is not to be used for long periods of time, disconnect the power cord to avoid damage from transient overvoltage.
- All cautions and warnings on the equipment should be noted.
- Please keep this equipment away from humidity.
- Do not use this equipment near water or a heat source.
- Place this equipment on a stable surface when installing. A drop or fall could cause injury.
- Never pour any liquid into the openings. This could cause fire or electrical shock.
- Openings in the case are provided for ventilation. Do not block or cover these openings. Make sure there is adequate space around the system for ventilation when setting up the

work area. Never insert objects of any kind into the ventilation openings.

- To avoid electrical shock, always unplug all power cords and modem cables from the wall outlets before removing covers.
- Lithium Battery provided (real time clock battery)

“CAUTION - Risk of explosion if battery is replaced by one of an incorrect type. Dispose of batteries according to instructions.”

- The equipment should be checked by service personnel if one of the following situation arises:
  - ▷ The power cord or plug is damaged.
  - ▷ Liquid has penetrated the equipment.
  - ▷ The equipment has been exposed to moisture.
  - ▷ The equipment is not functioning or does not function according to the user’s manual.
  - ▷ The equipment has been dropped and damaged.
  - ▷ The equipment has obvious signs of breakage.
- Never open the equipment. For safety reason, the equipment should only be opened by qualified service personnel.

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach a RMA application form which can be downloaded from: http://rma.adlinktech.com/policy/
2. All ADLINK products come with a two-year guarantee:
   - ▷ The warranty period starts from the product's shipment date from ADLINK's factory.
   - ▷ Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
   - ▷ For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for any loss of data.
   - ▷ Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
   - ▷ For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.

3. Our repair service is not covered by ADLINK's two-year guarantee in the following situations:
   - ▷ Damage caused by not following instructions in the user's manual.
   - ▷ Damage caused by carelessness on the user's part during product transportation.
   - ▷ Damage caused by fire, earthquakes, floods, lightning, pollution, other acts of God, and/or incorrect usage of voltage transformers.
   - ▷ Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals).
   - ▷ Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
   - ▷ Damage from improper repair by unauthorized technicians.
   - ▷ Products with altered and/or damaged serial numbers are not entitled to our service.
   - ▷ Other categories not protected under our warranty.
4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

If you have any further questions, please email our FAE staff: service@adlinktech.com.