EVA-X1610C

Development Kit

User's Manual

Acknowledgements

Part No. 2003161000

1st Edition Printed in Taiwan Feb. 2003

Product warranty

Advantech warrants to you, the original purchaser, that each of its products will be free from defects in materials and workmanship for one year from the date of purchase.

This warranty does not apply to any products that have been repaired or altered by persons other than repair personnel authorized by Advantech, or which have been subject to misuse, abuse, accident or improper installation. Advantech assumes no liability under the terms of this warranty as a consequence of such events.

Because of Advantech high quality-control standards and rigorous testing, most of our customers never need to use our repair service. If an Advantech product is defective, it will be repaired or replaced at no charge during the warranty period. For out-of-warranty repairs, you will be billed according to the cost of replacement materials, service time and freight. Please consult your dealer for more details. If you think you have a defective product, follow these steps:

1. Collect all the information about the problem encountered. (For example, CPU speed, Advantech products used, other hardware and software used, etc.) Note anything abnormal and list any on-screen messages you get when the problem occurs.
2. Call your dealer and describe the problem. Please have your manual, product, and any helpful information readily available.
3. If your product is diagnosed as defective, obtain an RMA (return merchandize authorization) number from your dealer. This allows us to process your return more quickly.
4. Carefully pack the defective product, a fully completed Repair and Replacement Order Card and a photocopy proof of purchase date (such as your sales receipt) in a shippable container. A product returned without proof of the purchase date is not eligible for warranty service.
5. Write the RMA number visibly on the outside of the package and ship it prepaid to your dealer.

Technical support and sales assistance

If you have any technical questions about the EVA-X1610C-DK & DK1 or any other Advantech products, please visit our support website at:

http://www.advantech.com/support

For more information about Advantech's products and sales information, please visit:

http://www.advantech.com

Contents

Figures

Tables

CHAPTER 1

Introduction

1.1 EVA-X1610C Description

The EVA-X1610C's design is based on a 16-bit, RISC architecture with a high performance microprocessor and compatibility of a 80C186 microprocessor, it also integrates the functions of SDRAM controller, non-multiplexed address bus, interrupt controller, DMA controller, timers, watchdog timer, FIFO UART serial ports, programmable I/O (PIO) pins and fast Ethernet MAC (Media Access Controller), all on chip.

1.2 EVA-X1610C Features

- 16-bit Microprocessor
  - RISC architecture with Pipeline Technology
  - Static design
  - 1 MB memory address space
  - 64 KB I/O address space
  - Software compatible with 80C186 microprocessor
  - Up to 100 MHz operating frequency
- SDRAM Control Interface
  - Support 1 Mbits x 16 and 4 Mbit x 16 SDRAM devices
  - Up to 100 MHz operating frequency
- One Non-Multiplexed System Bus Interface
  - Non-multiplexed address and data bus
  - 8-bit or 16-bit external bus dynamic access
  - Support Max. 512 KB, 8- or 16-bit data width Flash ROM
  - Multiplexed with SDRAM bus interface
  - Supports memory and I/O devices
- Supports an Independent Bus for I/O Devices
  - Multiplexed address and data
  - External latches needed to separate address and data bus
  - Only used for I/O devices to prevent from throttling the perfor mance of SDRAM & Non-Multiplexed System Bus
  - Share same pins as one of the FIFO UART channels
- Dual FIFO UART Serial Channels
  - Two Integrated high performance FIFO UART ports with 16 byte transmit/receive FIFO
  - Programmable Baud rate generator

  - Character options are programmable for start bit, stop bits, even/ odd or no parity, 5~8 data bits
  - One of the FIFO UART channels share same pins as independent I/O bus interface

- Fast Ethernet MAC Port with MII Interface
  - IEEE 802.3, 802.3u specification compliant
  - 10/100 Mbps data transfer rate
  - IEEE 802.3u compliant MII interface with serial management interface
  - Full-duplex/half duplex operation
  - Flow control for Full-duplex operations
  - VLAN Supported
  - Automatic CRC append and check
  - Multicast/Broadcast address recognition
  - Provided loop back path on MII interface
- Interrupt Controller
  - Seven maskable external interrupts provided
  - One non-maskable external interrupt
- Two Independent DMA Channels
- Three Independent 16-bit Timers
- One Independent Programmable Watchdog Timer
- Supports 16 PIO pins
- 3.3 V Operating Voltage
- 128 pins PQFP Package

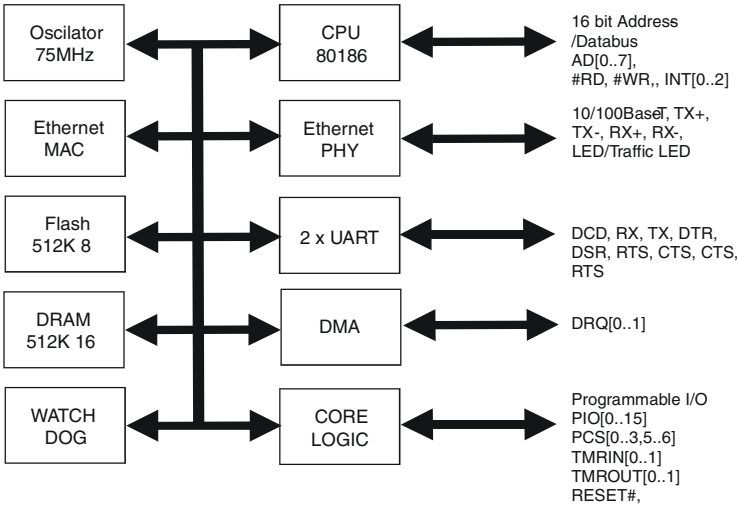


Fig 1-1 EVA-X1610C function block

NOTE

The EVA series related model name and its descriptions are listed below.

1. EVA-X1610C: 186-based Ethernet-enabled chipset.

2. EVA-SOM1610-01: 186-based Ethernet-enabled System on Module.

3. EVA-X1610C-DK: EVA-X1610C development kit without LCD panel.

4. EVA-X1610C-DK1: EVA-X1610C development kit with LCD panel.

1.3 EVA-X1610C Development Kit

EVA-X1610C provides two types of development kits, you can choosen either according to your requirements.

a. EVA-X1610C-DK: EVA-X1610C development kit without LCD panel.

b. EVA-X1610C-DK1: EVA-X1610C development kit with LCD panel.

The packing list for EVA-X1610C-DK and EVA-X1610C-DK1 are shown below:

Hardware

1. Paradigm C++ Advantech Edition (Box, including CD-ROM, Hardware License Key....etc.)
2. 186-based Ethernet-enabled System on Module (EVA-SOM1610-01).
3. EVA evaluation board without LCD panel. (EVA-X1610C-DK only)
4. EVA evaluation board with LCD panel. (EVA-X1610C-DK1 only)
5. 3-wired RS-232 Cable
6. Parallel Cable
7. AC adapter
8. Loop back communication connector x 2 pcs.
9. Plastic stud x 4 pcs

Software

1. Paradigm C++ Advantech Edition
2. EVA-A1610C development kit SDK CD
   i. ROM Monitor Demo Utility
   ii. NexGen TCP/IP Stacks
   - NexGenOS
   - NexGenIP
   - NexGenBOOT
   - NexGexRESOLV
   - NexGenMAIL
   - NexGenWEB
   - NexGenPPPoE
   - NexGenPPP

iii. EVA Software Development Kit

File name	Description
evainit.c	Hardware initialization
evais.asm	Interrupt/exception handling for EVA
nghwdefs.h	Hardware dependent definitions

- EVA Hardware Interface

File name	Description
evasio.c	UART standard I/O without interrupt handle
evaser.c	UART serial device driver
evamac.c	MAC Ethernet device driver with 4 RX descriptors and 1 TX descriptors
evamac.h	MAC Ethernet device hander file
evafmac.c	MAC Ethernet device driver with 8 RX descriptors and 4 TX descriptors
evafmac.h	MAC Ethernet device hander file
eva.h	Hardware registers definitions

- EVA Device Drivers (UART and MAC)

File name	Description
evautil.c evautil.h	CPU dependent utitlities, e.g. CPU bus, GPIO, Reset, etc

- EVA Utilities (CPU BUS, GPIO, Reset)

File name	Description
sstflash.c sstflash.h	SST Flash ROM control source code
evasys.c evasys.h	EVA system information source code
shc_flash.c shc_flash.h	Flash ROM embedded shell command
shc_sysctl.c shc_sysctl.c	EVA system embedded shell command
moncom.c moncom.h	ROM Monitor communication protocols source code
evamon.c	EVA ROM Monitor firmware main program
startup.asm	EVA ROM Monitor bootstrap source code
startup.inc	EVA ROM Monitor assembler utilities
locate.cfg	EVA ROM Monitor code/data segment configuration

- EVA ROM Monitor

Sample Proejct	File name
Microsoft VC++ 6.0	Moncomtest.dsw Moncomtes.dsp Moncomtest.c
Borland C++ Builder 5.0	PrjMonUtil.bpr PrjMonUtil.cpp PrjMonUtil.tds
	Monutil.cpp Monutil.dfm Monutil.h
	Trommon.cpp Trommon.h

iv. Others • ROM Monitor PC suite demo source codes

v. NexGenGRAPH

vi. LCD panel control driver

vii. Touch screen control driver

viii. Sample Programs for NexGenGRAPH

(v ~ viii, EVA-X1610C-DK1 only)

CHAPTER 2

Getting Started

Chapter 2 introduces how to build your development environment for the EVA-X1610C evaluation board, install related software tools and run a simple demo program.

2.1 EVA-X1610C Evaluation Board connection

Following are the steps that introduce how to connect your PC and EVA-X1610C evaluation board properly.

1. Connect EVA-X1610C evaluation board to the PC via a serial communication cable.

   Connect the 3-wire crossover serial communication cable to COM2 (UART1)[1] on the EVA-X1610C evaluation board. Connect the other end of the cable to a COM port on PC.

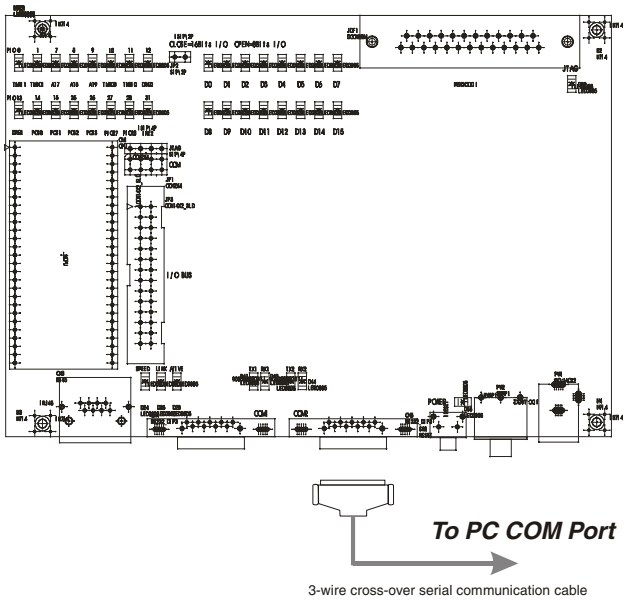


Fig. 2-1 Serial Communication Cable Connection

[1] In ROM Monitor Mode (Default setting), its console port is UART1. In non-ROM Monitor Mode, console port is usually UART0.

2. Connect the power supply.

Plug the end of power adapter into the jack PW1.

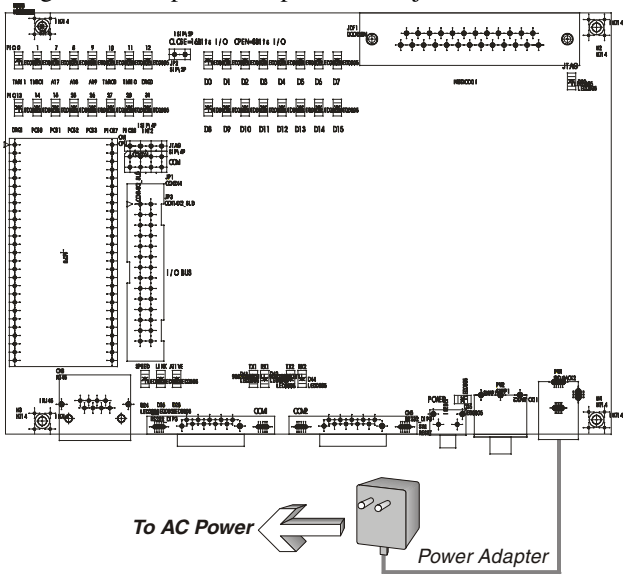
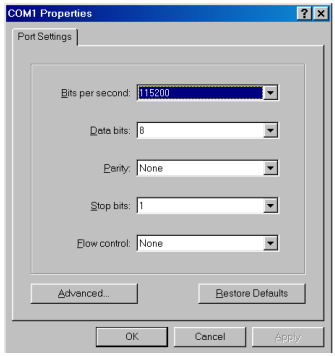


Fig. 2-2 Power Adapter Connection

3. Setup Hyper-Terminal

Press "**Start**" of task bar of window system, and then select "**Program**" → "**Accessories**" → "**Communication**" → "**Hyper-Terminal**".

For proper communication between the EVA-X1610C evaluation board and PC, the settings of Hyper-terminal must comply with the following:



- Connection :COM 1 (User defined)
- Baud Rate: 115200 bps
- Data bit: 8 bits
- Parity: None
- Stop bit: 1 bit
- Flow Control: None

4. Turn the EVA-X1610C evaluation board on.

   Switch the power (PW2) of the EVA-X1610C evaluation board on. If all the settings are right, the green LED on the EVA-SOM1610-01 will flicker and the Hyper-Terminal will show the prompt “>”.

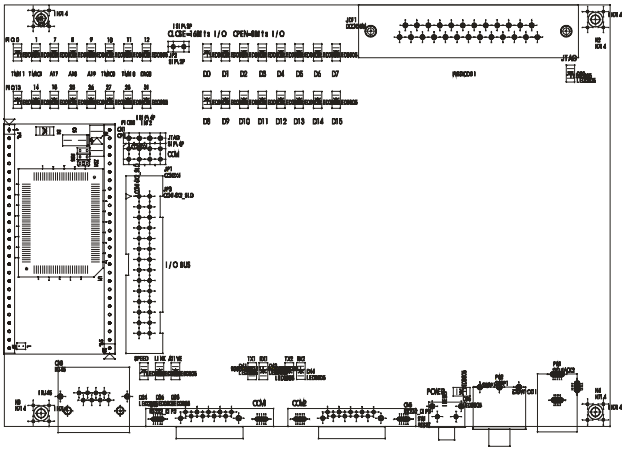


Fig 2-3 Powering on the EVA-X1610C Evaluation Board

NOTE:

There are three methods to reset the evaluation board.

1. Turn off the evaluation board via power switch (PW2).
2. Unplug the power adapter, then plug it back in.
3. Press the Reset button of the evaluation board.

2.2 Running Demo program

After you connect the EVA-X1610C evaluation board and PC properly, you may try running a demo program. First, execute “Hyper-Terminal” of windows and press “Enter” key. If the all connections are right, you will see the prompt “>” as shown below.

Enter the command “**help**”, you will see a list of available commands in this demo program as follows:

```
>help
Supported commands:
help        ver         exit        lsmod
netstat     ifconfig    arp         ping
exit        devstat     evamac      cls
dflush      dsetcolor   dsetfillcolor   dsettextcolor
dpixel      dline       dbox        dcircle
darc        dpolyline   getptchar   diag
sysctl      chksum
```

NOTE

1. To know the function of command, you can use the command “help”.

```
>help ver
Usage: ver
Display version string
```

2. To display information about command syntax, enter the command name directly.

```
>diag
Usage: diag [cmd [options]]
EVA-X1610C-DK1 diagnostic test commands.
[cmd [options]]:
        outportb [address(Hex)] [value(Hex)]
        inportb [address(Hex)]
        outportw [address(Hex)] [value(Hex)]
        inportw [address(Hex)]
        backlight [on/off]
        printts [on/off]
        adjts
        testts
        setgpio [pionum] [mode:0,X] [dir:0,X] [data:0,1,X(not set)]
        getgpio [pionum]
        testgpio
        testuart
        testall
```

Before you begin to use EVA-X1610C evaluation board, you must to configure the System Configuration. The example below shows how to display and configure the system configuration properly.

If you want to display the system configuration, you can use the command "**sysctl**".

```
>sysctl
version = 1.11.00                  /* System version*/
status = 1                         /* System status; 0: ROM-Monitor Mode.
                                   1: Application Mode */
arpaddr = 00000000                 /* ARP address */
id = EVA Chip                      /* Chip identification */
ethernet address = 000102-030405   /* MAC address */
subnet mask = 255.255.255.0        /* Subnet mask */
ip = 10.0.0.1                      /* EVA-X1610C IP */
gateway = 10.0.0.254               /* Gateway address */
aliveled = 0
```

[**Example**] Modify IP address

If you want to change IP address into 172.18.3.254, you can:

```
>sysctl ip 172.18.3.254
ip = 172.18.3.254
```

[**Example**] Modify Ethernet address

If you want to change the Ethernet address into 001122334456, you can:

```
>sysctl ethernet 00 11 22 33 44 56
ethernet address = 001122-334456
```

[**Example**] Modify system status

If you want to change system status into ROM-Monitor mode, you can:\

```
>sysctl status 0
status = 0
```

The system configuration will be available after you reboot the system

```
>sysctl reboot
Advantech EVA ROM Monitor/BIOS

Local IP address: 172.18.3.254
Ticks per second: 200

Wait for telnet connection at port 23
Wait for EVA ROM Moniter connection at port 7000
```

NOTE
After setting the IP and gateway IP of EVA-X1610C Evaluation board, you can connect the PC and EVA-X1610C Evaluation board by telnet.

To ensure the evaluation board is okay, you can use following commands to test your evaluation board.

- Diag testgpio — Test all of the General Purpose I/O ports.
- Diag testio — Test the I/O bits on the evaluation board.
- Diag testuart — Test COM1/COM2 (UART0/UART1) on the evaluation board.
- Diag testall — Test GPI/O, I/O, UART0, UART1 sequentially on the evaluation board.

NOTE
If you want to test a specific COM port, you must plug the loop back communication port on that port.

For example, the following are the results for execute the command "diag testall", the system will test GPI/O, I/O UART0 and UART1 sequentially.

NOTE
Remember to check the system status is in Application Mode. Otherwise set the status and reboot.

```
>diag testall
————————————[Backlight]————————————
Backlight turn on
————————————[GPIO]————————————
GPIO test tool.
Turn off all GPIO LEDs ...
Turn on GPIO LED one by one ...
[0][1][7][8][9][10][11][12][13][15][25][26][27][28][31]
Turn off GPIO LED one by one ...
[0][1][7][8][9][10][11][12][13][15][25][26][27][28][31]
Flash all GPIO LEDs 2 times
Finished.
————————————[UART]————————————
Cannot test UART1, please through telnet embedded shell !
————————————[Touch Screen]————————————
[Try 1] ...
Adjust touch screen...
Please click on the 1st cursor hot point....
Please click on the 2nd cursor hot point....
Please click on the 3rd cursor hot point for examing....
Finished.
————————————[Backlight]————————————
Backlight turn off
>
```

NOTE
Press the arrows on the LCD panel.

NOTE:

1. In this example, you can't test UART0 because UART0 is connecting with PC.
2. In order to test UART0, you can connect the PC and EVA-X1610C Evaluation board by telnet and test UART0 with the same command.

CHAPTER 3

Hardware Subsystems

3.1 EVA-X1610C

Fig 3-1 shows the block diagram for EVA-X1610C.

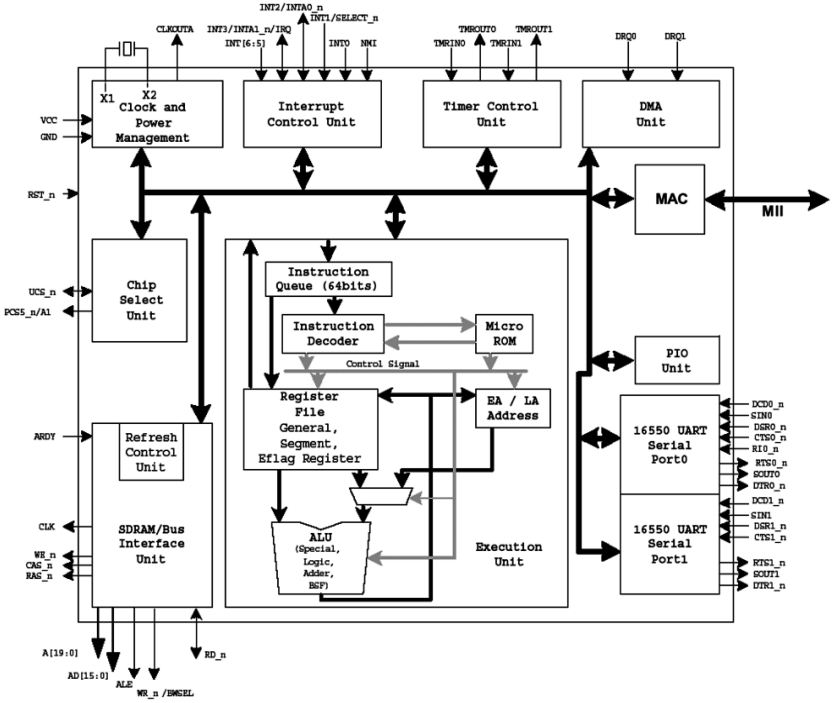


Fig 3-1 EVA-X1610C Block Diagram

3.2 EVA-SOM1610-01

The pin placements of EVA-SOM1610-01 are shown in Fig 3.2.

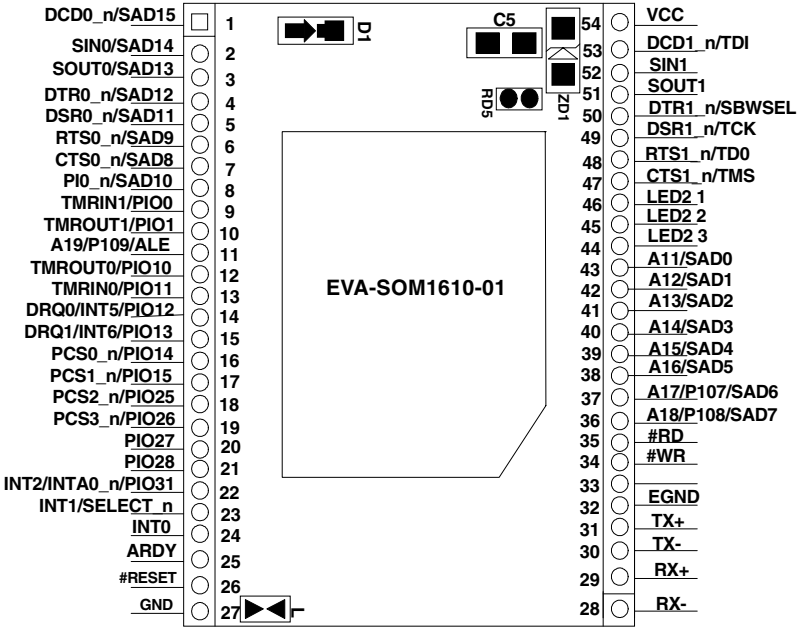


Fig 3.2 pin placement of EVA-SOM1610-01

EVA-SOM1610-01 Functional Description

I=Input; O=Output

CPU Core

PIN No.	Symbol	Type	Description
26	#RESET	I	Reset input with Switch Trigger. When RST_n is asserted, the CPU immediately terminates all operations, clears the internal registers and logic, and changes the address to the reset address FFFF0H.

Bus Interface

PIN No.	Symbol	Type	Description
35	#RD	O	Read Strobe. One active low signal indicates that the microcontroller is performing a memory or I/O read cycle. The RD_n floats during a bus hold or reset.
34	#WR	O	Write strobe. This pin indicates that the data on the bus is to be written into a memory or an I/O device. WR_n is active during T2, T3 and Tw of any write cycle, floating during a bus hold or reset. BWSEL is used to decide the boot ROM bus width when RST_n goes from low to high. If BWSEL is with an external pull-low resistor (10 Kohm), the boot ROM bus width is 8 bits. Otherwise the boot ROM bus width is 16 bits.
11 36 37 38 39 40 41 42 43	A19/PI09/ALE A18/PI08/SAD7 A17/PI07/SAD6 A16/SAD5 A15/SAD4 A14/SAD3 A13/SAD2 A12/SAD1 A11/SAD0	O/I	Address bus. Non-multiplexed memory or I/O addresses. The address bus is one-half of a SD_CLK period earlier than the AD bus. The address bus is in a high-impedance state during a bus hold or reset. SAD[7:0]:The combination pins with addresses and data. They are designed for slower peripheral bus. ALE: Address latch enable. Active high. This pin indicates an address output an the AD bus. Address is guaranteed to be valid on the trailing edge of ALE.

Chip Select Unit Interface

PIN No.	Symbol	Type	Description
17 16	PCS1_n/PIO15 PCS0_n/PIO14	O/I	Peripheral chip select. These pins are active low when the micro-controller accesses the defined peripheral memory block (I/O or memory address). For I/O access, the base address can be programmed in the region from 0000H to 0FFFFH. For memory address access, the base address can be located in the 1 MB memory address region. These pins assert with the multiplexed AD address bus and do not float during bus holds.
18 19	PCS2_n/PIO25 PCS3_n/PIO26	I/O	Peripheral chip select. These pins are active low when the microcontroller accesses the defined peripheral memory block (I/O or memory address). For I/O access, the base address can be programmed in the region from 0000FH to 0FFFFH. For memory address access, the base address can be located in the 1 MB memory address region. These pins match with the multiplexed AD address bus and do not float during bus holds.

Interrupt Control Unit Interface

PIN No.	Symbol	Type	Description
22	PIO31	I/O	Maskable Interrupt Request 2/Interrupt Acknowledge 0. For INT2, it's active high. The interrupt input can be configured as either edge-triggered or level-triggered. The requesting device must hold the INT2 until the request is acknowledged to guarantee interrupt recognition. For INTA0_n, in cascade mode or special fully-nested mode, this pin corresponds to the INT0.
23	INT1/SELECT_n	I	Maskable Interrupt Request 1/slave select. For INT1, except for the differences in the interrupt line and interrupt address vector, the function of INT1 is the same as that of INT2. For the SELECT_n feature, when the microcontroller is used as a slave device, this pin is driven from the master interrupt controller decoding. This pin is activated to indicate that an interrupt appears on the address and data bus. The INT0 must be activated before the SELECT_n is activated when the interrupt type appears on the bus.
24	INT0	I	Maskable interrupt request 0. Except for differences in the interrupt line and interrupt address vector, the function of INT0 is the same as that of INT2.

Timer Control Unit Interface

PIN No.	Symbol	Type	Description
9 13	TMRIN1/PIO0 TMRIN0/PIO11	I/O	Timer input. These pins can be used as clock or control signal input, depending upon the programmed timer mode. After internally synchronizihg low to high transitions on TMRIN, the timer controller increments. These pins must be pulled up if not being used.
10 12	TMROUT1/PIO1 TMROUT0/PIO10	O/I	Timer output. Depending on timer mode select. These pins provide single pulse continuous waveform. The duty cycle of the waveform is programmable. These pins float during a bus hold or reset.

DMA Unit Interface

PIN No.	Symbol	Type	Description
15 14	DRQ1/INT6/PIO13 DRQ0/INT5/PIO12	I/O	DMA request. These pins are asserted high by an external device when the device is ready for DMA channel 1 or channel 0 to perform a transfer. These pins are level-triggered and internally synchronized. The DRQ signals must remain active until finished being serviced and not latched. For INT6/INT5: When the DMA function is not used, the INT6 and INT5 can be used as an additional external interrupt request. They share the corresponding interrupt type and register control bits. The INT6/5 are edge-triggered only and must be held until the interrupt is acknowledged.

High Speed UART

PIN No.	Symbol	Type	Description
2	SIN0/SAD14	I/O	SIN0: Serial Input. Serial Data Input from the communications link. SAD14: The combination pin with Address and Data. It is for slower device bus.
3	SOUT0/SAD13	I/O	SOUT0: Serial Output. Composite serial data output to the communications link. SAD13: The combination pin with Address and Data. It is for slower device bus.
6	RTS0_n/SAD9	I/O	RTS0_n: Request To Sand. When low, this indicates to MODEM or data set that URAT is ready to exchange data. SAD9: The combination pin with Address and Data. It is for slower device bus.

PIN No.	Symbol	Type	Description
4	DTR0_n/SAD12	I/O	DTR0_n: Data Terminal Ready. When low, this informs the MODEM or data set that UART is ready to establish a communication link. SAD12: The combination pin with Address and Data. It is for slower device bus.
7	CTS0_n/SAD8	I/O	CTS0_n:Clear To Send. When low, this informs that MODEM or data set is ready to exchange data. SAD8: The combination pin with Address and Data. It is for slower device bus.
5	DSR0_n/SAD11	I/O	DSR0_n: Data Set Ready. When low, this is indicates that MODEM or data set is ready to establish the communication link with UART. SAD11: The combination pin with Address and Data. It is for slower device bus.
1	DCD0_n/SAD15	I/O	DCD0_n: Data Carry Detection. When low, it indicates that the data carrier has been detected by the MODEM or data set. SAD15: The combination pin with Address and Data. It is for slower device bus.
8	PI0_n/SAD10	I/O	PI0_n: Ring Indicator. This indicates that a telephone ringing signal has been received by the MODEM or data set. SAD10: The combination pin with Address and Data. It is for slower device bus.
52	SIN1	I	SIN1: Serial Data Input.
51	SOUT1	O	SOUT1: Serial Data Output.
48	RTS1_n/TDO	O	RTS1_n: Request To Send. TDO: JTAG test data output pin.
50	DTR1_n/SBWS-EL	I/O	DTR1_n:Data Terminal Ready. SBWSEL is to decide the SAD bus width when the RST_n pin goes from low to high. If SBWSEL is with a pull-low resistor (10k ohm), the SAD bus width is 8 bits and 16550's Port 1 is active, Otherwise the SAD bus width is 16 bits and 16550 Port 1 is inactive.
47	CTS1_n/TMS	I	CTS1_n: Clear To Send. JTAG Test mode select.
49	DSR1_n/TCK	I	DSR1_n: Data Set Ready. TCK: JTAG test reset input
53	DCD1_n/TDI	I	DCD1_n: Carry Sense Detection TDI: JTAG test data input port.

GPIO Interface

PIN No.	Symbol	Type	Description
20	PIO27	I/O	General purpose PIN.
21	PIO28	I/O	General purpose PIN.

Ethernet Port

PIN No.	Symbol	Type	Description
47	LED1	O	Ethernet LED • Speed LED
46	LED2	O	Ethernet LED • Link LED
45	LED3	O	Ethernet LED • Active LED
28	Rx	I	Ethernet signal (Rx-)
29	Rx+	I	Ethernet signal (Rx+)
30	Tx-	O	Ethernet signal (Tx-)
31	Tx+	O	Ethernet signal (Tx+)
32	EGND		Ethernet ground.

3.3 EVA-X1610C Evaluation Board

Fig 3.3 shows the layout for EVA-X1610C evaluation board. It includes 16 GPIO LEDs, 16 I/O LEDs, one Ethernet port, one JTAG port for troubleshooting and two COM ports for communication.

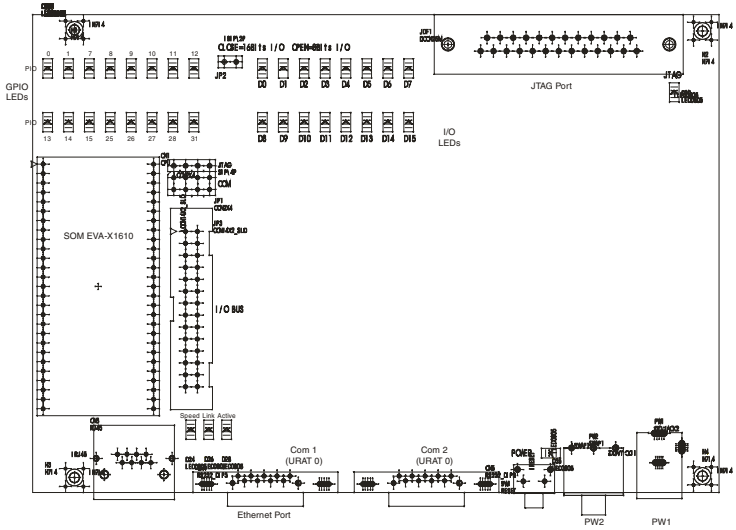


Fig 3.3 Layout for EVA-X1610C Evaluation board

3.4 Ethernet Networking

EVA-X1610C provides one Ethernet port. You can send or receive data between PC and EVA-X1610C evaluation board via Ethernet. The Ethernet pin assignment is shown in Fig 3.4.

There are three LEDs on the evaluation board that indicates the Ethernet status (Speed, Link, Active).

Pin	Signal Name
1	Tx +
2	Tx -
3	Rx+
4	None
5	None
6	**Rx -**
7	None
8	Non

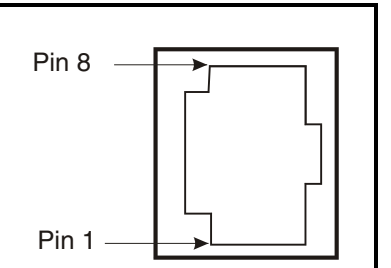


Fig 3.4 Ethernet port pin assignment

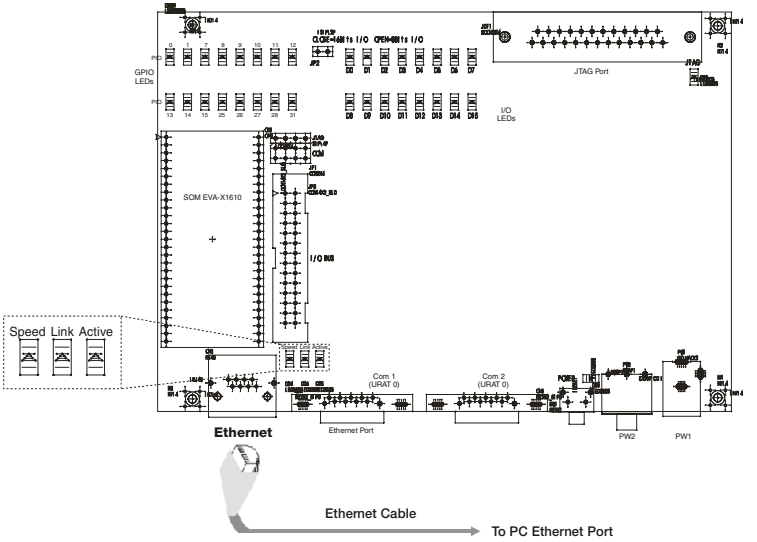


Fig 3.5 Ethernet port on the EVA-X1610C evaluation board

3.5 General-Purpose I/O

EVA-X1610C provides 16 programmable I/O signals, which are multi-functional pins with other signals of normal functions. Software must be used to configure these multi-functional pins as PIOs or normal functions by means of programming through these registers (7Ah, 78h, 76h, 74h, 72h, and 70h). Those LEDs indicate 16 general purpose I/O status (PIO 0, 1, 7, 8, 9, 10, 11, 12, 13, 14, 15, 25,26, 27, 28, 31). (Table 3.1).

Table 3-1 PIO-Multi-functional Pin list table

PIO No.	Pin No. (PQFP)	Multi Function	Reset Status/PIO internal registe
0	9	TMRIN1	PIO/ Input with 75K pull-up
1	11	TMROUT1	PIO/ Input with 75K pull-down
7	46	A17/SAD6	Normal operation/ Input with 75K pull-up
8	44	A18/SAD7	Normal operation/ Input with 75K pull-up
9	43	A19/ALE	Normal operation/ Input with 75K pull-up
10	12	TMROUT0	PIO/ Input with 75K pull-down
11	10	TMRIN0	PIO/ Input with 75K pull-up
12	14	DRQ0/INT5	PIO/ Input with 75K pull-up
13	13	DRQ1/INT6	PIO/ Input with 75K pull-up
14	126	PCS0_n	PIO/ Input with 75K pull-up
15	125	PCS1_n	PIO/ Input with 75K pull-up
25	34	PCS2_n	PIO/ Input with 75K pull-up
26	35	PCS3_n	PIO/ Input with 75K pull-up
27	39		PIO/ Input with 75K pull-up
28	41		PIO/ Input with 75K pull-up
31	5	INT2/INTA0_n	PIO/ Input with 75K pull-up

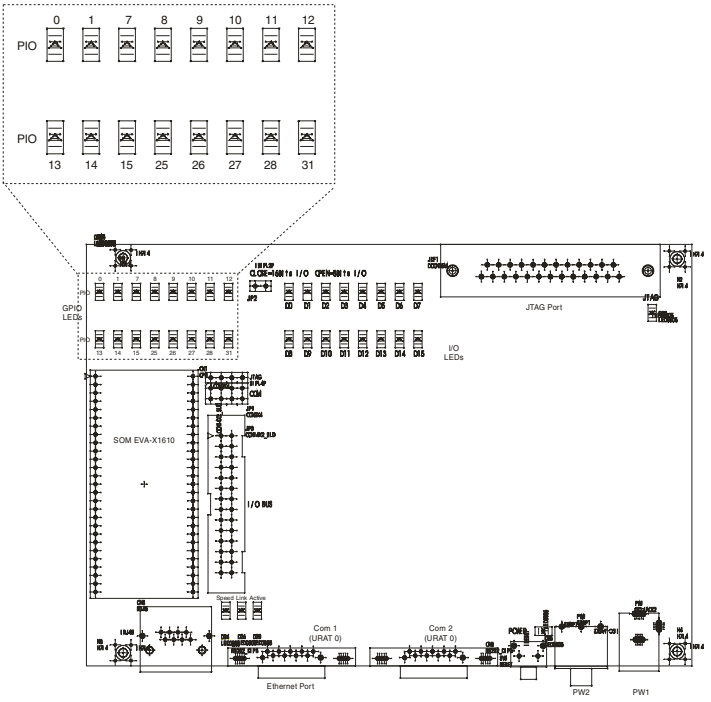


Fig3.6 The GPIO on the evaluation board

3.6 Data Bus

EVA-X1610C provides 16 data bits, D0~D15, which is indicated via 16 LEDs on the evaluation board. You can configure the data bus for 8 bits or 16 bits by JP2. The configuration of JP2 is shown in table 3.2. The default data bus configuration is 8-bit.

NOTE

If you configure the data bus for 16 bits by JP2, the UART0 will be disabled.

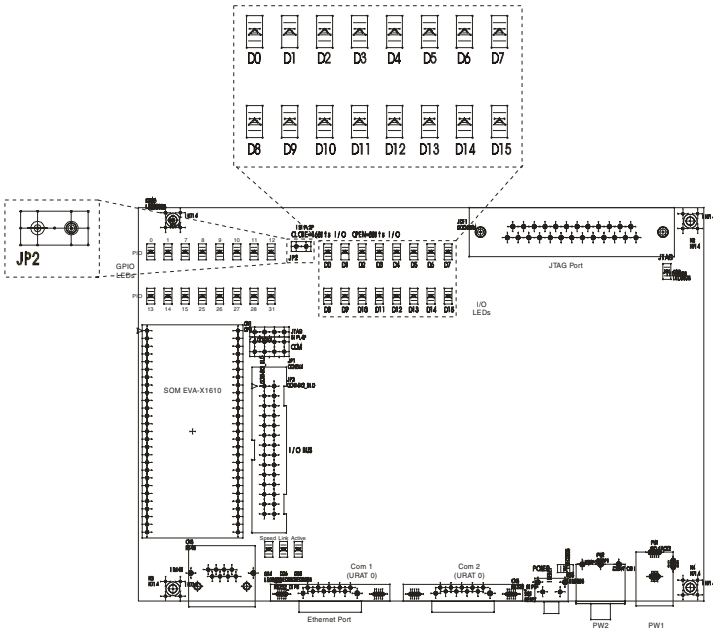
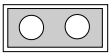
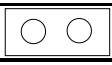


Fig 3.7 Location of data bits (D0~D15)

Table 3.2 The JP2configuration

JP2	Description
	Closed. The data bus is 16-bit. (D0~D15). And UART0 will be disabled.
	Open. The data bus is 8-bit (D0~D7).

3.7 I/O Bus

EVA-X1610C provides an I/O bus that is separate from EVA-SOM1610-01. The bus is a common set of wires that allows you to connect the EVA-X1610C evaluation board and your devices together.

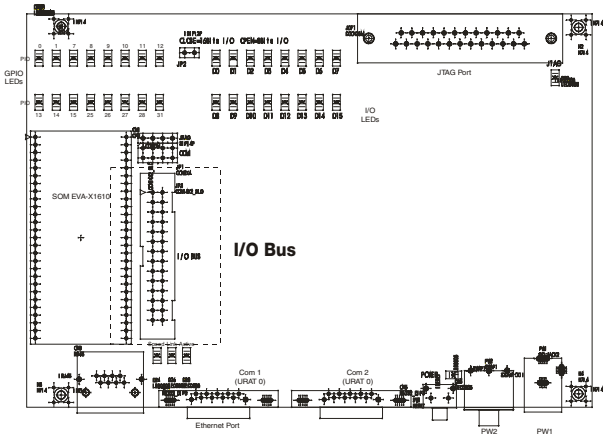


Fig 3.8 The location of the I/O bus

The pin placements of the I/O bus are shown as Fig 3.9. For further information about pin functionality of the I/O bus, please refer to section 3.2.

GND	1	2	VCC
TMRIN1	3	4	SAD0
TMROUT1	5	6	SAD1
TMROUT2	7	8	SAD2
TMRIN0	9	10	SAD3
DRQ0	11	12	SAD4
DRQ1	13	14	SAD5
#PCS0	15	16	SAD6
#PCS1	17	18	SAD7
#PCS2	19	20	#RD
#PCS3	21	22	#WR
INT2	23	24	ALE
INT1	25	26	ARDY
INT0	27	28	#RESET

I/O BUS

Fig 3.9 Pin placement of I/O bus

3.8 Serial Communication

EVA-1610C has two RS-232 serial communication interfaces (UART0, UART1). You can use a serial communication cable that connects a PC to the EVA-X1610C evaluation board. Simply plug one end of the cable into the serial port on PC and plug the other end into the serial port jack on your EVA-1610C evaluation board.

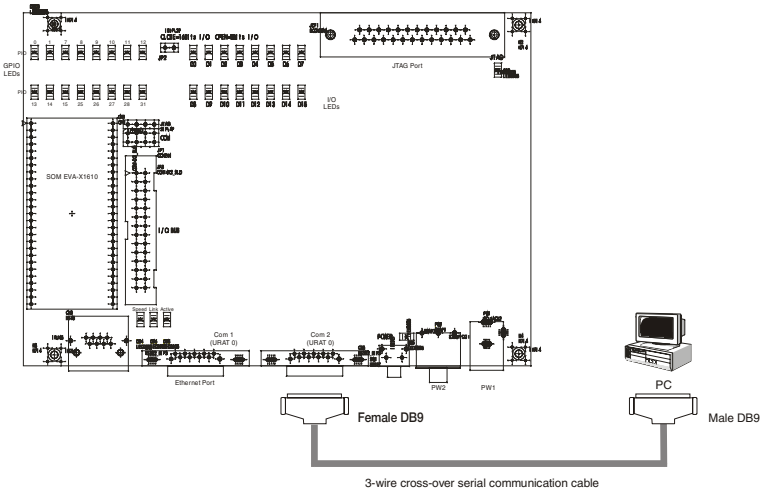


Fig 3.10 Connection between PC and EVA-X1610C Development Kit

You can configure JTAG and COM2 (UART1) via jumper JP1 and the RD5 on SOM module.

Table 3.3 JP1 and RD5 Configuration

JP1 / RD5	JTAG Port / COM 2		JTAG Port / COM 2	
	JTAG Port	COM 2	JTAG	COM 2
Open	Enable	Disable	Disable	Enable (Flow control function disable)
Close	Disable	Disable	Disable	Enable

The pin assignment of the RS-232 is shown as follows:

Pin No.	Description
Pin 1	DCD
Pin 2	Rx
Pin 3	Tx
Pin 4	DTR
Pin 5	GND
Pin 6	**DBR**
Pin 7	**RTS**
Pin 8	CTS
Pin 9	RI

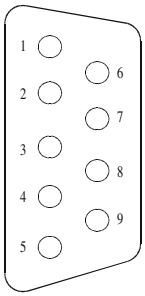


Fig 3.11 COM1 (UART0) pin assignment

Pin No.	Description
Pin 1	DCD
Pin 2	Rx
Pin 3	Tx
Pin 4	DTR
Pin 5	GND
Pin 6	DBR
Pin 7	RTS
Pin 8	CTS
Pin 9	None

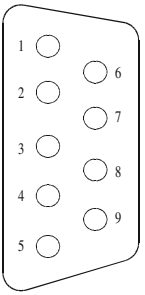


Fig 3.12 COM2 (UART1) pin assignment

3.9 Programming/Debugging Port (JTAG parallel port)

EVA-1610C evaluation board has a programming/debugging port that is designated JTAG parallel port. Users can download or debug custom programs via the JTAG parallel port. Fig 3.13 shows the location of the JTAG parallel port. When downloading or debugging programs, the LED (D20) near the JTAG parallel port will flicker.

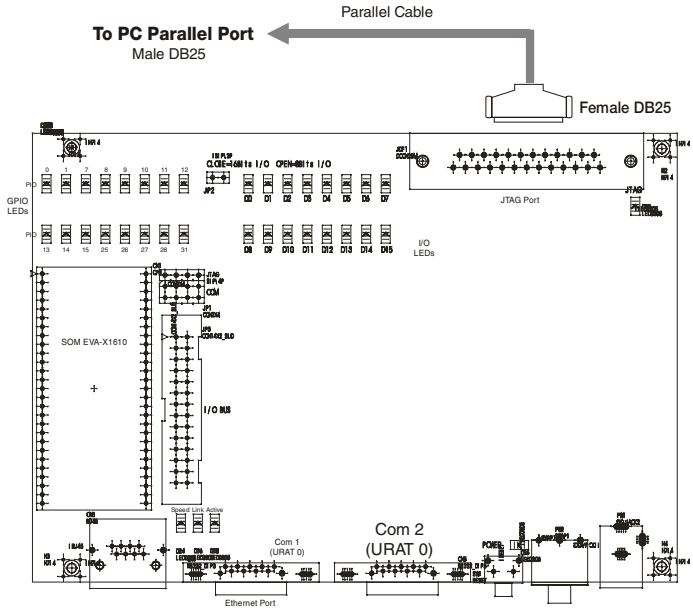


Fig 3.13 Connection between EVA-X1610C evaluation board and PC

3.10 LCD Display (option)

The EVA-X1610C evaluation board with LCD panel (EVA-X1610C-DK1) is shown as Fig 3-14:

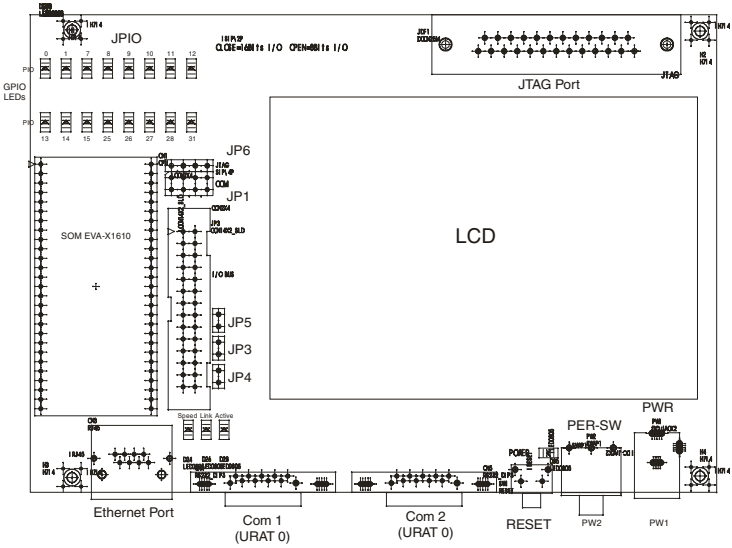
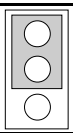
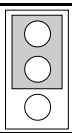
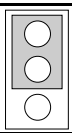
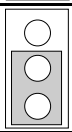
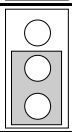
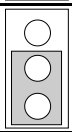


Fig 3-14 Layout for EVA-X1610C-DK1

The LCD in EVA-X1610C-DK1 supports touchscreen functions. If you want to enable touch screen functionality for a LCD panel, you can configure it by setting jumper JP3, JP4 and JP5 as table 3.4.

Table 3.4 JP3, JP4, JP5 Configurationngdhcps.h

JP3	JP4	JP5	Function
			Touchscreen enable COM1 (UART0) disable
			Touchscreen disable COM1 (UART0) enabl

CHAPTER 4

Software

4.1 Software Development Tools

EVA-X1610C development kit provides two software development tools.

- Paradigm C++ Advantech Edition
- ROM Monitor Demo Utility

4.1.1 Paradigm C++ Advantech Edition

This session introduce how to install and configure Paradigm C++ Advantech Edition.

4.1.1.1 Install Paradigm C++ Advantech Edition

Step1: Insert Paradigm C++ Advantech Edition CD into your CD-ROM (autorun) or execute Setup.exe in Paradigm C++ Advantech Edition CD.

Step2: After execute Setup.exe, it will show the dialog window as below, press Next> button to continue installation.

Fig 4.1 Welcom

Step3: Check if there is a USB Hardware License Key inside the box. It is essential to keep installing the software. Press Next> button to continue installation.

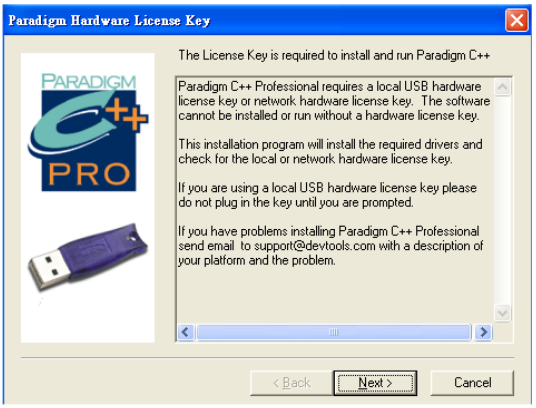


Fig 4.2 Paradigm Hardware License Key

Step4: Insert the USB Hardware License Key, your OS will find its driver. Select "Local USB hardware license key" and press Next> button to continue.

Fig 4.3 Select the type of license key

Step5: Enter your Name, Organization and the product serial number found on the CD jacket or Paradigm license agreement.

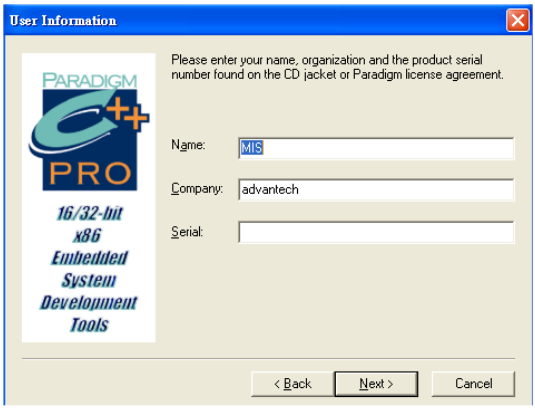


Fig 4.4 User Information

Step6: Read the Paradigm license agreement. To install Paradigm C++, you must accept this agreement.

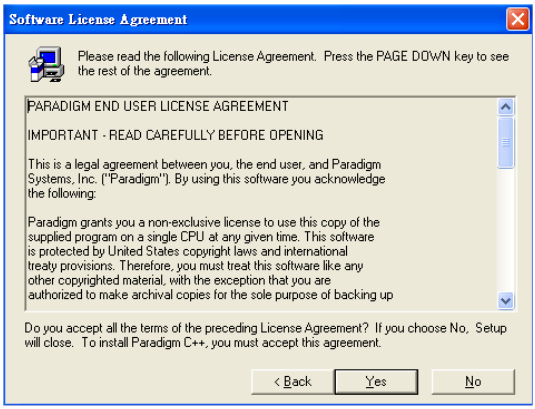


Fig 4.5 Software License Agreement

Step7: Select Destination Folder.The default directory for installation is “C:\Program Files\Paradigm”. If user wants to change the directory, press Browse ... button and select the folder for installation. After selecting the folder for installation, press Next> button to continue.

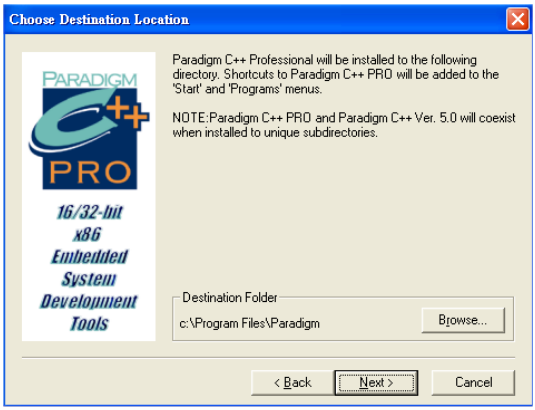


Fig 4.6 Choose Destination Location

Step8: Select the real mode install options to install. After selection, press Next> button to continue.

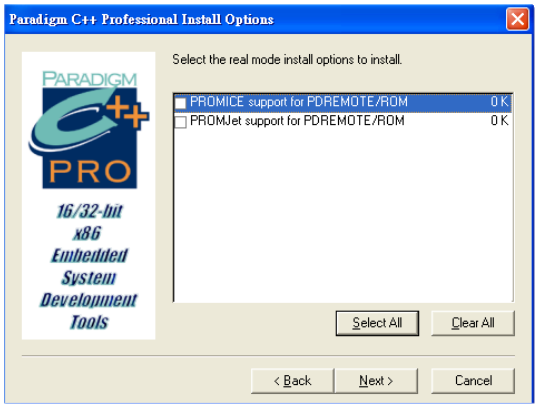


Fig 4.7 Install options

Step9: If you want to review or change any settings, click <Back> . If not, click Next> to continue.

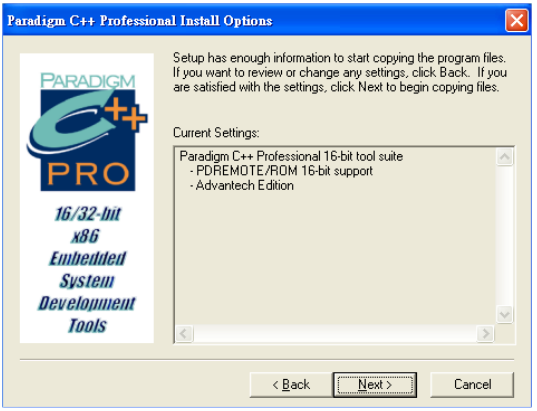


Fig 4.8 Current Settings Review

Step10: Installing Paradigm C++ Advantech Edition

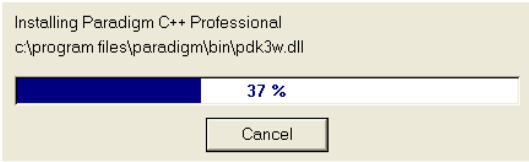


Fig 4.9 Installing

Step11: Setup complete. Press Finish button to finish the installation of Paradigm C++ Advantech Edition.

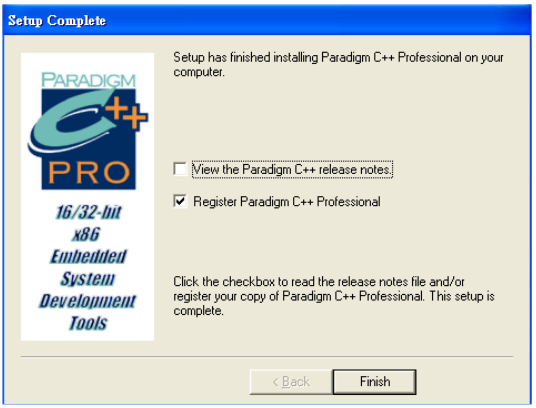


Fig 4.10 Setup Complete

Step12: Restart your computer.

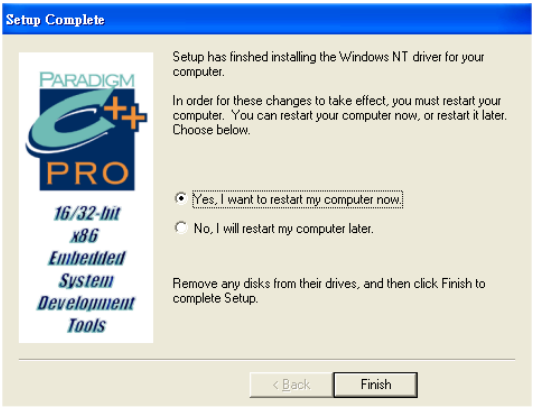


Fig 4.11 Restart your computer

Notice:

1. After setup, there will be an IE window (http://www.devtools.com/register/form.asp) shows that you must register to qualify for 90 days technical support. You can register this page in order to get updates. But because this program is free for evaluation purposes, license and technical support is only available for 30days free license. If you do have questions, please check out the Paradigm C++ Advantech Edition help files, electronic manuals, and newsgroups for more information. See the Help menu of the demo software. A Quick start guide in .PDF format, located in Start|Programs|Paradigm C++ Advantech Edition|Docs can walk you through the basics of Paradigm C++ Advantech Edition.

2. After 30 days trial period, you can cantact Paradigm to get software license. URL: http://www.devtools.com/contact.htm, or E-MAIL: sales@devtools.com

3. Everytime before using Paradigm C++ Advantech Edition, be sure to put the Hardware License Key into your computer.

4.1.1.2 Download demo program

Step1: Firstly, execute Start|Programs|Paradigm C++ Advantech Edition|Paradigm C++. Insert the Paradigm Hardware License Key.

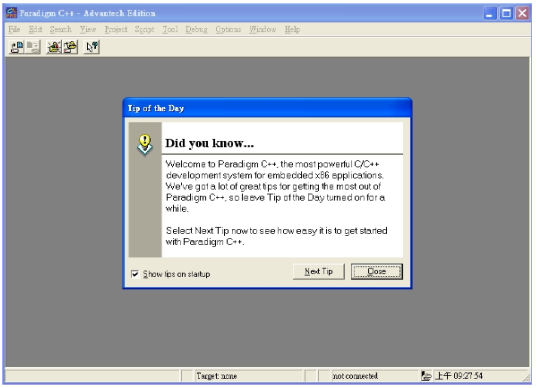


Fig 4.12 Execute Paradigm C++ Advantech Edition

Step2: Connect the development kit with your computer properly (Please refer Chapter 2). And connect the JTAG cable between development kit and computer.

Step3: Select Project|Open Project... in pull-down menu and choose an IDE file that you want to download to the firmware of EVA-X1610. You may try the IDE file in the SDK UNZIP PATH\adveva\i186\apps\evamon\ideui directory.

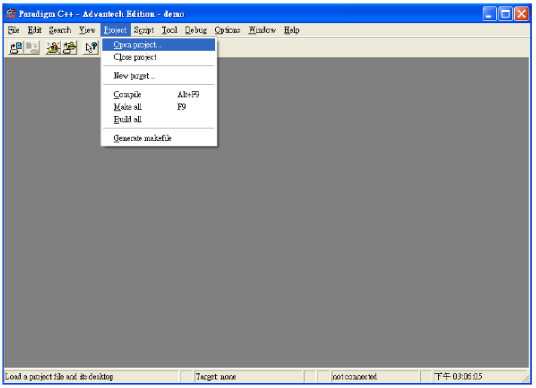


Fig 4.13 Open a project

Step4: Select Debug|Run in pull-down menu to download the program to the firmfare.

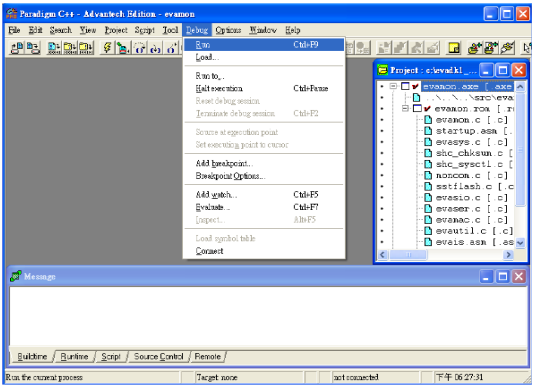


Fig 4.14 Download program

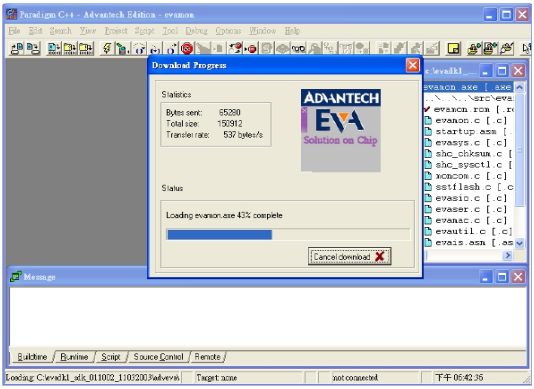


Fig 4.15 Downloading

For more information about install, execute and uninstall Paradigm C++ Advantech Edition Development Tool, please refer to the documents in Start|Programs|Paradigm C++ Advantech Edition|Docs.

4.1.2 ROM Monitor Demo Utility

When applications have been developed and fully debugged, then the projects can integrate ROM Monitor related resources with a few lines modification. ROM Monitor has been taken care the most complex things about Compiler, CPU BUS, Flash ROM, SDRAM, and communication protocols. And make your applications be runtime upgradeable in a few minutes thru Ethernet if you need it. Once you choose this mode, Paradigm C++ Advantech Edition or Flash Burner is no longer an only way for application firmware download. Before you can download application firmware to target, make sure the ROM Monitor firmware (in the SDK UNZIP PATH\adveva\i186\apps\evamon\ideui directory) has been burned into Flash ROM first by burner.

4.1.2.1 Execute ROM Monitor Demo Utility

Step1: execute ROM Monitor Demo Utility

Execute ROM Monitor Demo Utility as the following dictionary: SDK UNZIP PATH\adveva\tests\win32\moncom\monutil\ PrjMonUtil.exe

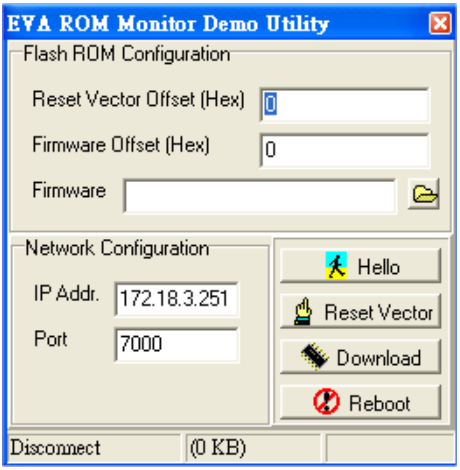


Fig 4.16 ROM Monitor Demo Utility

Step2: Set vector and firmware offset
Set vector and firmware offset to 0

Step3: Network Configuration
Set network configuration value of EVA-X1610C-DK/DK1 in the Network Configuration block.
[Example]
IP Addr: 172.18.3.254 (IP of EVA-X1610C, User defined)
(You can modify IP address by the command "sysctl")
1 Port: 7000 (Fixed value)

Step4: Confirm the connection between EVA-X1610C and PC properly.
Connect EVA-X1610C-DK/DK1 to Ethernet. Click 

button. If the connection between the PC and EVA-X1610C evaluation board is correct, it will show the message as below on your hyperterminal screen.

```
ROM Monitor: accept connection from address: 172.18.3.29:1205
ROM Monitor: [Hello A]
ROM Monitor: [Offline]
ROM Monitor: connection closed.
ROM Monitor: servicing new connection...
```

Step5: Load the firmware file.

Click  button, choose the firmware ROM Binary file to download to the flash ROM in EVA-X1610C evaluation board.

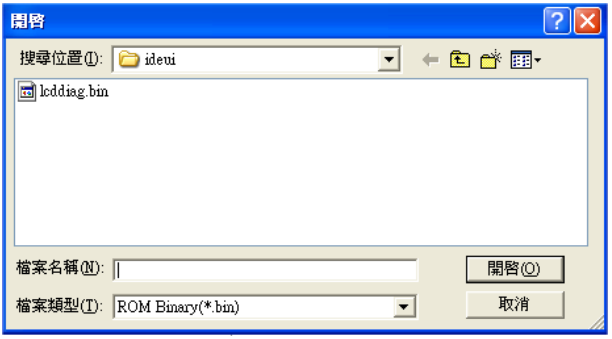


Fig 4.17 Download the firmware file

Step6: Download

Click Download button and it will download the firmware ROM Binary file to the Flash ROM in EVA-X1610C evaluation board.

Step7: Reset Vector

Click Reset Vector button. System control will be transferred from ROM Monitor to the application.

Step8: Reboot

Click Reboot button to restart the system. If all the settings are right, EVA-X1610 will execute the specific application that you downloaded before.

NOTICE:

Next time you want to download the firmware ROM Binary file to the Flash ROM by ROM Monitor Demo Utility, remember to set status to 0 via telnet or hyper terminal before downloading (Using the command "sysctl", please refer to session 2.2).

4.2 Function APIs

4.2.1 EVA Utility

Source: evautil.c

Header file: evautil.h

1. AdvBusNormal()

Description	set CPU BUS mode to normal
Parameters	None
Return Value	None

2. AdvBusShadow()

Description	Move code from ROM to RAM and then set CPU BUS mode to Shadow mode
Parameters	srcSeg: source segment, 0x0008, 0x000c, etc dstSeg: destination segment, 0x0008, 0x000c, etc len: length of code to be moved, 64KByte based. 1 means 64KByte
Return Value	None

3. AdvResetSystem()

Description	Reset system by enabling watchdog
Parameters	None
Return Value	None

1. AdvGpioGetAddr()

Description	Get relative GPIO bit index and related registers address
Parameters	pioNum: GPIO number (0 ~ 31) pio: → index: GPIO bit index → mode.reg: GPIO mode register address → dir.reg: GPIO direction register address → data.reg: GPIO data register address
Return Value	**0: success** **Otherwise: fail**

2. AdvGpioOutput()

Description	Set GPIO to (1,0) and then pullup or pulldown data bit
Parameters	pio: GPIO address get by AdvGpioGetAddr() upDown: GPIO data bit enable or disable
Return Value	0: success Otherwise: fail

3. AdvGpioInput()

Description	Set GPIO to (0, 1) and then read data bit
Parameters	pio: GPIO address get by AdvGpioGetAddr() upDown: GPIO data bit up(1) or down(0)
Return Value	0: success Otherwise: fail

4. AdvGpioModeDirDataSet()

Description	Set GPIO mode, direction and data
Parameters	pio: GPIO address get by AdvGpioGetAddr() mode: GPIO mode; (0: set to 0 ; others: set to 1) dir: GPIO direction; (0: set to 0 ; others: set to 1) data: GPIO data; (0: set to 0 ; 1: set to 1 ; other: not set)
Return Value	0: success Otherwise: fail

5. AdvGpioModeDirDataGet()

Description	Get GPIO mode, direction and data
Parameters	pio: GPIO address get by AdvGpioGetAddr() mode: GPIO mode (0 or 1) dir: GPIO direction (0 or 1) data: GPIO data (0 or 1)
Return Value	0: success Otherwise: fail

4.2.2 EVA ROM Monitor

The main purpose of ROM monitor is to assist you to download and upgrade the firmware. Compared with Paradigm C++ Advantech Edition and flash burner, ROM monitor can upgrade your application in a few minutes. The weakness of ROM monitor is that you can't debug your application via Paradigm C++ Advantech Edition in this mode. Therefore, ROM monitor is suitable for applications that have been developed and fully debugged.

Fig 4.18 shows the architecture of EVA ROM monitor. When you start up the EVA-X1610C, it will follow below procedures:

(1) ROM monitor is loaded from flash ROM to the lower address of SDRAM.

(2) If the system is in ROM Monitor Mode (status=0), it will proceed to initialize hardware (uCOS kernel, serial device, Ethernet, etc.)

If the system is in Applcation Mode (status=1), AP will be loaded to SDRAM and booted up.

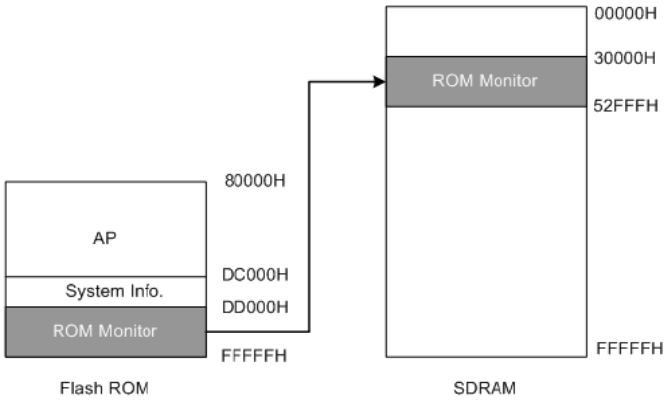


Fig 4.18 the architecture of EVA ROM monitor

When the system is in the ROM Monitor mode (status=0), you can upgrade your firmware in this mode. The upgrade proceduces is shown as Fig 4.19.

(1) If you want to upgrade the Flash ROM, AP will send a “Download” request to ROM monitor. If ROM accept this request, it will response an ACK to AP, otherwire, ROM monitor will response an NACK to AP and reject this request.

(2) If ROM monitor accept a download request from AP, AP will start to update the Flash ROM. AP will send a “Update” request to ROM Monitor. This request includes header and data and ROM monitor will check the header. If all information are right, ROM monitor will response an ACK and then write the information to Flash ROM.

(3) When AP finishes to upgrage the Flash ROM, AP will send a “Final” request to ROM monitor. If ROM accept this request, it will response an ACK to AP, otherwire, ROM monitor will response a error message to AP.

(4) If you want to execute the application that you downloaded, you must set system status to 1 and then reboot the system.

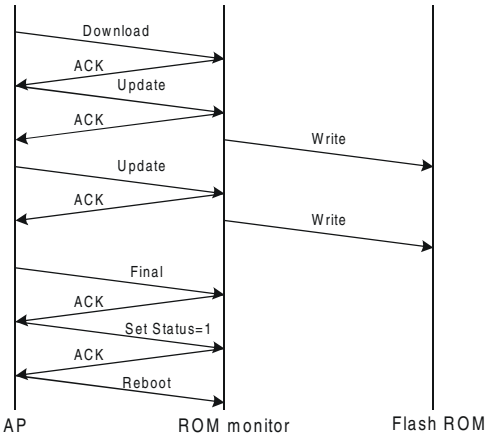


Fig 4.19 Upgrade firmware procedures

If your application wants to be downloaded via ROM monitor, you have to follow below steps:

(1) Include header files of ROM monitor in your application1.

```
#include <shc_flash.h>
#include <shc_sysctl.h>
#include <moncom.h>
```

(2) Add the below paragraph in the embedded shell of application.

```
/*
 * Embedded shell
 */
    :
    :
 &advShellCmd_sysctl,
```

(3) Add the below paragraph before your main routine of application.

```
void systemInfo(const NGcfgent *oldcfg, int len)
{
  int ret;
   NGcfgent cfg, newconfig[sizeof(myconfig)/
sizeof(myconfig[0])];
  EVA_SYSTEM_INFO evainfo={ /* default system set-
ting */
   EVA_SYSTEM_INFO_STATUS_ROMMONITOR,
    0x0L,
   EVA_SYSTEM_INFO_VERSION,
    "EVA Chip",
    {
      {
        NET_ETHADDR,
        NET_ADDR,
        NET_NETMASK,
        IP_GATEWAY
      },
```

```
      {0}
   }};

  /* read EVA system information from flash */
             if(AdvEvaSystemInfo(&evainfo,
EVA_SYSTEM_INFO_OVERWRITE_NEW) < 0) /* if new flash
then overwrite flash with evainfo */
  {
   ngPrintf("Fail to fetch Flash ROM type !\n");
   return;
  }

  /* copy default configuration data to new array */
  ngMemCpy(newconfig, oldcfg, sizeof(oldcfg[0])*len);

  /* make some changes for configuration */
  cfg.cfg_option = NG_ETHIFO_ADDR; /* MAC0 ethernet
address */
  cfg.cfg_arg = NG_CFG_PTR( evainfo.mac[0].macAddr);
  AdvEvaAlterConfig(newconfig, len, 0, &cfg);
  cfg.cfg_option = NG_IFO_ADDR; /* MAC0 IP address
*/
  cfg.cfg_arg = evainfo.mac[0].ipAddr;
  AdvEvaAlterConfig(newconfig, len, 0, &cfg);
  cfg.cfg_option = NG_IFO_NETMASK; /* MAC0 netmask
*/
  cfg.cfg_arg = evainfo.mac[0].netmask;
  AdvEvaAlterConfig(newconfig, len, 0, &cfg);
   cfg.cfg_option = NG_IPO_ROUTE_DEFAULT; /* MAC0
gateway IP address */
  cfg.cfg_arg = evainfo.mac[0].gateway;
  AdvEvaAlterConfig(newconfig, len, 0, &cfg);

  /* initialise stack */
  if( (ret = ngInit( newconfig)) != NG_EOK) {
   ngPrintf( "can't initialize stack, error=%04X\n",
ret);
    ngExit(1);
```

(4) Paragraph B insteads of paragraph A as below in your main routine of application

Paragraph A (Original)

```
  /* initialise stack */
  if( (i = ngInit( myconfig)) != NG_EOK) {
     ngPrintf( "NexGenOS: can't initialize stack,
error=%04X\n", i);
    ngExit(1);
  }
```

Paragraph B (Revised)

```
systemInfo( myconfig, sizeof(myconfig)/
sizeof(myconfig[0]));
```

(5) Insert below paragraph in your MAKE files.

In the following example is the paragraph of MAKE files of iptest1 that indicates how to revise your make file properly.

```
# Directories
      :
      :
ADVEVADIR = $(subst \,/,$(ADVEVA))
```

```
# Compiler and linker flags
:
:
include $(ADVEVADIR)/i186/make/pcc.mk
```

```
# EVA Utilities
:
:
# SST flash files
OBJS += shc_flash.$(NGOEXT) sstflash.$(NGOEXT)
```

```
# EVA system info files
OBJS += evasys.$(NGOEXT)

# EVA system control files
OBJS += shc_sysctl.$(NGOEXT)
```

```
# source files location
:
:
$(ADVEVADIR)/i186/src:$(ADVEVADIR)/i186/src/
evamon:$(ADVEVADIR)/i186/src/samples
:
:
locate $(LOCATEFLAGS) -c$(ADVEVADIR)/i186/src/sam-
ples/locate.cfg $(PROGNAME).rom
```

The source codes of EVA-X1610C ROM monitor are listed below:

- ROM Monitor Firmware Source Codes
  Souce: evamon.c (EVA ROM monitor firmware main program)
  startup.asm (EVA ROM monitor bootstrap source code)
  startup.inc (EVA ROM monitor assembler utilies)
  locate.cfg (EVA ROM monitor code and data segment configruation)
- Flash ROM Control Source Codes
  Source: sstflash.c
  Header file: sstflash.h

1. AdvSSTFlashID()

Description	Get Flash ROM manufactor and device ID
Parameters	flashRom: Flash ROM Context structure pointer segment: Any effective Flash ROM address
Return Value	0

3. AdvSSTFlashSectorErase()

Description	Erase one sector of Flash ROM
Parameters	flashRom: Flash ROM Context structure pointer offset: Start address (offset related to the Flash ROM lowest address) of sector to be erased
Return Value	None

4. AdvSSTFlashByteProgram()

Description	Program one byte to Flash ROM
Parameters	flashRom: Flash ROM Context structure pointer offset: Start address (offset related to the Flash ROM lowest address) the data byte to be written to. data: data byte to be programmed to Flash ROM
Return Value	0: success 1: failure

5. AdvFlashInitialize()

Description	Initialize Flash ROM Context structure, including get Flash ROM ID, assign Flash ROM subroutines and change CPU BUS to normal mode if necessary.
Parameters	flashRom: Flash ROM Context structure pointer itrCtl: **FLASH_ROM_NO_CRITICAL_ACCESS:** Without critical section control in all Flash ROM subroutines. Caller must call critical section routine obviously before call this function. **FLASH_ROM_CRITICAL_ACCESS:** With critical section control in all Flash ROM subroutines. Caller must not call critical section routine before call this function.
Return Value	0: success 1: failure

6. AdvFlashTerminate()

Description	Change CPU BUS to shadow mode if necessary
Parameters	flashRom: Flash ROM Context structure pointer
Return Value	None

- EVA System Information Source Codes

  Source: evasys.c

  Header file: evasys.h

1. AdvEvaSystemInfo()

Description	Get set EVA system information from/to Flash ROM
Parameters	info: which store default informations of EVA. overwrite: EVA_SYSTEM_INFO_OVERWRITE_NONE: don't overwrite flash, just read EVA_SYSTEM_INFO_OVERWRITE_FORCE: force overwrite flash EVA_SYSTEM_INFO_OVERWRITE_NEW: overwrite only when new flash EVA_SYSTEM_INFO_OVERWRITE_DEVID: overwrite only when different devID
Return Value	0: write information to flash 1: read information from flash -1: if unknown Flash ROM, unknown overwrite flag or error

- Embedded Shell Source Codes?Flash ROM embedded shell command
  Source: shc_flash.c
  Header file: shc_flash.h

Function API name: advShellCmd_flashctl()

EVA system embedded shell command "flashctl". This command provides Flash ROM low-level control. Its syntax is shown below.

flashctl [cmd [options]]

cmd	Description
id	Product identification.
ease offset	Erase flash ROM sector; erase flash must be multiples of 4KB1.
reset	Reboot system.
pogram offset byte	Write a byte in a specificed offset. This specificed offset must be erased first.

- Embedded Shell Source Codes?EVA system embedded shell command
  Source: shc_sysctl.c
  Header file: shc_sysctl.h

Function API name: advShellCmd_sysctl()

EVA system embedded shell command “sysctl”. This command can display and modify system configuration. Its syntax is shown below:

sysctl [cmd [options]]

cmd	Description
version	System version, this parameter can’t be modified.
status	System Mode Status=0; ROM-Monitor Mode. Status=1; Application Mode.
arpaddr	ARP address
id	Product identification.
ethernet address	MAC address
subnet mask	Subnet mask
Ip	EVA-X1610C IP
gateway	Gateway addresss
reboot	Reboot system.

- ROM Monitor Communication Protocols Source Codes
  Source: moncom.c
  Header file: moncom.h

1. AdvRomMonInit()

Description	Initialize ROM Monitor Communication Context
Parameters	ctx: pointer of ROM Monitor Communication Context should not be a NULL pointer or point to NULL sk: connected socket, suitable for both of client site and server site
Return Value	Always EVA_ROMMON_ERR_NONE

2. AdvRomMonRecv()

Description	Receiving package from peer. This routine will response necessary acknowledgements to peer. Possible acknowledgements would be, 1. EVA_ROMMON_COMMAND_HELLOB 2. EVA_ROMMON_COMMAND_ACK 3. EVA_ROMMON_COMMAND_NACK
Parameters	ctx: pointer of ROM Monitor Communication Context should not be a NULL pointer or point to NULL
Return Value	Return: EVA_ROMMON_ERR_NONE EVA_ROMMON_ERR_UNKNOWN EVA_ROMMON_ERR_SOCKET EVA_ROMMON_ERR_TIMEOUT EVA_ROMMON_ERR_INVALID_LEN EVA_ROMMON_ERR_INVALID_PACKET EVA_ROMMON_ERR_INVALID_CMD EVA_ROMMON_ERR_UNCOMPLETE EVA_ROMMON_ERR_CHECKSUM

3. AdvRomMonSend()

Description	sending package to peer. This routine will expect necessary acknowledgements from peer. Possible acknowledgements would be, 1. EVA_ROMMON_COMMAND_HELLOB 2. EVA_ROMMON_COMMAND_ACK 3. EVA_ROMMON_COMMAND_NACK
Parameters	ctx: pointer of ROM Monitor Communication Context should not be a NULL pointer or point to NULL
Return Value	Return: EVA_ROMMON_ERR_NONE EVA_ROMMON_ERR_UNKNOWN EVA_ROMMON_ERR_SOCKET EVA_ROMMON_ERR_TIMEOUT EVA_ROMMON_ERR_INVALID_LEN EVA_ROMMON_ERR_INVALID_PACKET EVA_ROMMON_ERR_INVALID_CMD EVA_ROMMON_ERR_UNCOMPLETE EVA_ROMMON_ERR_CHECKSUM

4. AdvRomMonFinal()

Description	Cleanup ROM Monitor Communication Context
Parameters	ctx: pointer of ROM Monitor Communication Context should not be a NULL pointer or point to NULL
Return Value	Always EVA_ROMMON_ERR_NONE

5. AdvRomMonCmdSet()

Description	Set and Prepare ROM Monitor Command Package
Parameters	ctx: pointer of ROM Monitor Communication Context should not be a NULL pointer or point to NULL cmd: ROM Monitor Commands EVA_ROMMON_COMMAND_HELLOA EVA_ROMMON_COMMAND_HELLOB EVA_ROMMON_COMMAND_ACK EVA_ROMMON_COMMAND_NACK EVA_ROMMON_COMMAND_REBOOT EVA_ROMMON_COMMAND_DOWNLOAD_INIT EVA_ROMMON_COMMAND_DOWNLOAD_UPDATE EVA_ROMMON_COMMAND_DOWNLOAD_FINAL EVA_ROMMON_COMMAND_OFFLINE
Return Value	EVA_ROMMON_ERR_NONE EVA_ROMMON_ERR_INVALID_LEN EVA_ROMMON_ERR_INVALID_CMD

4.3 Sample programs

All sample programs are provided in the EVA-X1610C CD-ROM, all software development kit were be compressed into a .zip file, thus, you must uncompress this file first. The paths of sample programs are shown below:

ROM Monitor and Sample Programs

Program Name	Sample code directory	Sample configuration and make files directories
The following samples are for both of EVA-X1610C-DK and EVA-X1610C-DK1		
ROM Monitor*	$(ADVEVA)/i186/src/evamon	$(ADVEVA)/i186/apps/evamon
EBDIAG(	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/ebdiag
IPTEST	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/iptest
VFSTEST	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/vfstest
PPPCLI	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/pppcli
PPPSERV	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/pppserv
PPPOETST	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/pppoetst
The following samples are EVA-X1610C-DK1 only		
LCDDIAG*	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/lcddiag
D_BUFFER	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/d_buffer
DRVTEST	$(ADVEVA)/i186/src/samples	$(ADVEVA)/i186/apps/samples/drvtest

NexGen Software [*without ROM Monitor, download by DT or burner*]

	Sample code directory	Sample configuration and make files directories
NexGenOS	$(NGOS)/tests/gentests	$(NGOS)/tests/ucosii/i186/eva/os $(NGOS)/tests/ucosii/i186/eva/serial
NexGenIP	$(NGIP)/tests/gentests	$(NGIP)/tests/ucosii/i186/eva/iptest
NexGenBOOT	$(NGBOOT)/tests/gentests	$(NGBOOT)/tests/ucosii/i186/eva/dhcpc
NexGenRESOLV	$(NGRESOLV)/tests/gentests	$(NGRESOLV)/tests/ucosii/i186/eva/test1
NexGenMAIL	$(NGMAIL)/tests/gentests	$(NGMAIL)/tests/ucosii/i186/eva/smtp
NexGenWEB	$(NGWEB)/tests/gentests	$(NGWEB)/tests/ucosii/i186/eva/httpc $(NGWEB)/tests/ucosii/i186/eva/vfs
NexGenPPPoE	$(NGPPPOE)/tests/gentests	$(NGPPPOE)/tests/ucosii/i186/eva/pppoecli
NexGenGRAPH	$(NGGRAPH)/tests/gentests	$(NGGRAPH)/tests/ucosii/i186/eva/d_buffer $(NGGRAPH)/tests/ucosii/i186/eva/drvtest
NexGenPPP	$(NGPPP)/tests/gentests	$(NGPPP)/tests/ucosii/i186/eva/pppserv

[Example] Running ***ROM Monitor*** sample program

If you connect EVA-X1610C evaluation board UART1 properly to PC with NULL modem cable, you will see boot up message on the Hyper-terminal. This shipped default program in Flash ROM is ROM Monitor with diagnostic program. And the active partition is diagnostic program. To change active partition to ROM Monitor, please type shell command "sysctl status 0" and followed "sysctl reboot" command from console (or from telnet channel). If you want the diagnostic program back, just type shell command "sysctl status 1" and followed "sysctl reboot". You can also burn ROM Binary file of this program by burner from below path (Please notice that once you hard burn ROM Binary file to Flash ROM, the diagnostic selftest program will be erased. That means you could not enter shell command "sysctl status 1" again, unless another application under ROM Monitor mode has been download by utility already):

ROM Monitor's make file directionary is shown as below:

$(ADVEVA)/i186/apps/evamon

ROM Monitor includes the shell commands presented below. You can use HELP to display the list of shell command.

Standard Commands

Command	Description
help	Display list of commands and help information
ver	Display version of NexGenOS.
netstat	Display information about the network drivers and protocols
devstat	Display information about the configuration of installed serial devices and their statistics.
ifconfig	Display information and configures network interfaces
sysctl	Display and configure system configuration.
route	Display routing table.
lsmod	Display list of loaded module in ROM Monitor sample program.
exit	Terminate the shell session.

Test Commands

Command	Description
ping	Sends ICMP echo requests to a host. This command can be used to verify the IP communication between two hosts.
arp	Display Address Resolution Protocol host table.
diag	Diagnose general purpose GPIO, I/O, UART port in the EVA-X1610C evaluation board.

CHAPTER 5

Real-Time Operating System

5.1 MicroC/OS-II

EVA-X1610C's real time operating system kernel is MicroC/OS-II. It is a highly portable, ROMable, very scalable, preemptive real-time, multitasking kernel (RTOS) for microprocessors and microcontrollers. Below are the features of MicroC/OS-II:

Portable: Most of MicroC/OS-II is written in highly portable ANSI C, with taget microprocessor-specific code written in assemly language. Assembly langeage is kept to a minimum to make MicroC/OS-II easy to port other processors.

ROMable: MicroC/OS-II was designed for embedded applications. Thus, You can embed MicroC/OS-II into your specific application.

Preemptive: MicroC/OS-II is a fully preemptive real-time kernel. It means that MicroC/OS-II always runs the highest priority task that is ready.

Multitasking: MicroC/OS-II can manage up to 64 tasks; however the current version of the software reserves eight of these tasks for system use.

Deterministic: Execution time of all MicroC/OS-II funtions and services and deterministic. Except for pme service, executeion time of all MicroC/OS-II services don't depend on the number of tasks running in the application.

Task Stacks: Each task require its own stack; however, MicroC/OS-II allows each task to have a different stack size. This allows users to reduce the amount of RAM needed in the application.

MicroC/OS-II can manage up to 63 application tasks and provides the following services:

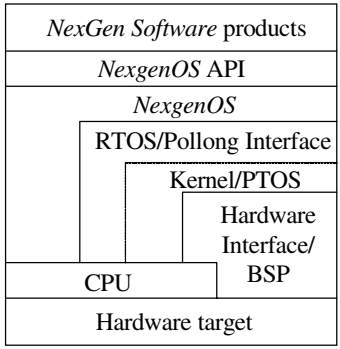
- Semaphores
- Event Flags
- Mutual Exclusion Semaphores (to reduce priority inversions)
- Message Mailboxes
- Message Queues
- Task Management (Create, Delete, Change Priority, Suspend/Resume etc.)
- Fixed Sized Memory Block management
- Time Management

5.2 NexGenOS

NexGexOS is a low-level layer that isolates all CPU, OS and compiler dependencies. The aim of NexGenOS is to provide all necessary functions to help write a platform independent application. For this purpose, the following five classes of services are implemented:

- Low Level Abstraction layer that encapsulates both the hardware and the operating system dependencies
- Standard initialisation mechanism for installing drivers and network protocols
- Device driver manager that gives better control of the different devices
- Shell utility to be used as a basis for all testing activities
- Highly optimised standard library to be used in embedded applications

An application using the NexGenOS architecture is totally portable and does not rely on any particular hardware or operating system. It is even possible to avoid using an operating system altogether. All NexGen Software products have been written for NexGenOS and are consequently totally platform independent. The figure 5.1 below shows the different layers of NexGenOS.



NexGenOS layers

Fig 5.1: NexGenOS Layers

The NexGenOS API is summarized in the figure below. NexGenOS is able to run in multi-tasking mode (when used on top of a RTOS) and in polling mode (without OS), the API being the same in both modes and only minor changes are necessary in the application structure. The task management and synchronisation routines are of course only available in multi-tasking mode.

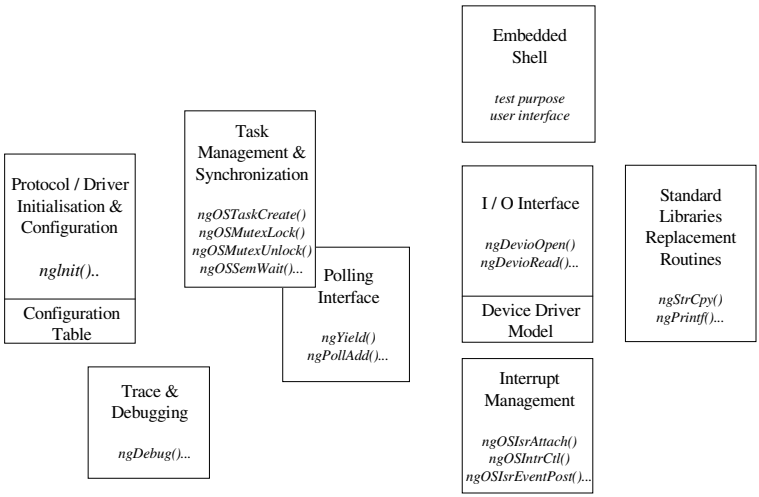


Fig 5.2 NexGenOS API

Further information about NexGenOS, please refer to the documentation “NexGexOS Pguide_V1_3_20011030”.

5.3 NexGenOS APIs

API name	Header	Parameter	Return value	Remark
ngATOH	ngos.h	str	Converted value	Converts a string to an unsigned 32-bits integer.
ngATOI	ngos.h	str	Converted value	Converts a string to an integer.
ngATOL	ngos.h	str	Converted value	Converts a string to a long integer.
ngBETOH16, ngBETOH32	ngos.h	value	The converted value in host representation	Converts big endian values to host representation.
ngBSwap16, ngBSwap32	ngos.h	value	The byte-swapped value	Swaps bytes of a value.
ngDebug	ngos.h	nodule, submod, level, format	None	Displays debug information.
ngDebugSetLevel	ngos.h	level	None	Sets debug level.
ngDebugSetModule	ngos.h	module, onoff	None	Enables or disables debugging for a specific module.
ngDevioClose	ngos.h ngdev.h	iop	0	Closes a serial device.
ngDevioInChar	ngos.h ngdev.h	iop	A positive or NULL character value if the call was successful, otherwise a negative error code	Reads one character from a serial device.
ngDevioIoctl	ngos.h ngdev.h	Iop, level, cmd, arg, arglen	0	Sets or gets options on the device.
ngDevioGetNRead	ngos.h ngdev.h	Iop	0	Gets the number of bytes in the device input buffer.
ngDevioGetState	ngos.h ngdev.h	Iop	The state of the device if the call was successful, otherwise a negative error code	Gets the state of the device.
ngDevioOpen	ngos.h ngdev.h	devp, flags, iop	0	Opens a serial device for basic I/O operations.
ngDevioOutChar	ngos.h ngdev.h	c, iop	None	Writes one character to a serial device
ngDevioPrintf	ngos.h ngdev.h	Iop, fmt	The number of bytes sent if the call was successful, otherwise a negative error code	Writes formatted text to a device.
ngDevioRead	ngos.h ngdev.h	Iop, buf, buflen, flags,	The number of bytes read if the call was successful, otherwise a negative error code	Reads data from a device.
ngDevioReadByte	ngos.h ngdev.h	Iop, flags	The character read if the call was successful, otherwise a negative error code	Reads one byte from a device.

API name	Header	Parameter	Return value	Remark
ngDevioReadEx	ngos.h ngdev.h	Iop, buf, buflen, min, time	The number of bytes read if the call was successful, otherwise a negative error code	Reads data from a device (extended version).
ngDevioWrite	ngos.h ngdev.h	Iop, buf, buflen, flags	The number of bytes written if the call was successful, otherwise a negative error code	Writes data to a device.
ngDevioVPrintf	ngos.h ngdev.h	Iop, fmt, args	The number of bytes sent if the call was successful, otherwise a negative error code	Writes formatted text to a device.
ngDevioVPrintf	ngos.h ngdev.h	Iop, fmt, args	The number of bytes sent if the call was successful, otherwise a negative error code	Writes formatted text to a device.
ngExit	ngos.h	status	This function does not return to its caller	Terminates the program.
ngGetChar	ngos.h	none	The character read if the call was successful, otherwise a negative error code	Reads one character from standard input.
ngGetVersionString	ngos.h	none	A pointer to the version string	Gets product version string.
ngHTOBE16, ngHTOBE32	ngos.h	value	The converted value in big endian representation	Converts host values to big endian representation.
ngHTOLE16, ngHTOLE32	ngos.h	value	The converted value in little endian representation	Converts host values to little endian representation.
ngInit	ngos.h	conftable	0	Initialises NexGenOS.
ngLETOH16, ngLETOH32	ngos.h	value	The converted value in host representation	Converts little endian values to host representation.
ngMemBlkAlloc	ngos.h ngmem-blk.h	memp	A pointer to the allocated block or NULL if no free block was available	Allocates a fixed-size memory block.
ngMemBlkFree	ngos.h ngmem-blk.h	memp, ptr	None	Releases a fixed-size memory block.
ngMemBlkInit	ngos.h ngmem-blk.h	memp, addr, blk_nb, blk_size	NG_EOK	Initialises a pool of fixed-size memory blocks.
NG_MEMBLK_-ALLOC	ngos.h ngmem-blk.h	memp, ptr	None	Allocates a fixed-size memory block (macro).
NG_MEMBLK_F-REE	ngos.h ngmem-blk.h	memp, ptr	None	Releases a fixed-size memory block (macro).

API name	Header	Parameter	Return value	Remark
ngMemChr	ngos.h	s, c, len	A pointer to the character or NULL if the character is not found	Locates the first occurrence of a character in a buffer.
ngMemCmp	ngos.h	s1, s2, len	Integer	Compares two memory objects.
ngMemCpy	ngos.h	dst, src, len	A pointer to the destination buffer	Copies non-overlapping memory objects.
ngMemMove	ngos.h	dst, src, len	A pointer to the destination buffer	Copies possibly overlapping memory objects.
ngMemSet	ngos.h	dst, c, len	A pointer to the destination buffer	Fills memory with a constant byte.
ngOSClockGetFreq	ngos.h	none	The number of clock ticks per second	Gets the frequency of the system clock.
ngOSClockGetTime	ngos.h	none	Value of the system clock	Returns the system clock value.
ngOSGetErrno	ngos.h	none	The error code of the last system call	Returns the error code of the last system call.
ngOSIntrCtl	ngos.h	ctl	The previous interrupt state	Controls interrupts.
ngOSIsrAttach	ngos.h	Irq, handler_f, data	A positive or NULL value if the call was successful, otherwise a negative error code	Installs an interrupt handler routine.
ngOSIsrDetach	ngos.h	Irq	A positive or NULL value if the call was successful, otherwise a negative error code	Removes an interrupt handler routine.
ngOSIsrEventDelete	ngos.h	event	NG_EOK	Deletes an event.
ngOSIsrEventGetSize	ngos.h	none	The size of the NGOSievent structure	Gets the size of the interrupt event descriptor structure.
ngOSIsrEventInit	ngos.h	event	NG_EOK	Initialises an interrupt event.
ngOSIsrEventPost	ngos.h	event	NG_EOK	Triggers an event from an interrupt handler.
ngOSIsrEventTaskPost	ngos.h	event	NG_EOK	Triggers an event from a task.
ngOSIsrEventWait	ngos.h	event	NG_EOK	Waits for an event.
ngOSMemKTOM	ngos.h	ptr, addr	NG_EOK	Gets the physical address associated with a pointer.
ngOSMemMTOK	ngos.h	addr, len, ptr	NG_EOK	Gets a pointer to a 32-bits physical address.
ngOSMutexDelete	ngos.h	mutex	NG_EOK	Deletes a mutex.
ngOSMutexGetSize	ngos.h	none	The size of the NGOSmutex structure	Gets the size of the mutex descriptor structure.

API name	Header	Parameter	Return value	Remark
ngOSMutexInit	ngos.h	mutex	NG_EOK	Initialises a mutex.
ngOSMutexLock	ngos.h	mutex	NG_EOK	Locks a mutex.
ngOSMutexUnlock	ngos.h	mutex	NG_EOK	Unlocks a mutex.
ngOSSemClear	ngos.h	seg	NG_EOK	Clears the semaphore count.
ngOSSemDelete	ngos.h	seg	NG_EOK	Deletes a semaphore.
ngOSSemGetSize	ngos.h	none	The size of the NGOSsem structure	Gets the size of the semaphore descriptor structure.
ngOSSemInit	ngos.h	sem, value	NG_EOK	Initialises a semaphore.
ngOSSemPost	ngos.h	sem	NG_EOK	Increments a semaphore.
ngOSSemTimedWait	ngos.h	sem, dly	NG_EOK	Waits on a semaphore with a timeout.
ngOSSemWait	ngos.h	sem	NG_EOK	Waits on a semaphore
ngOSSetErrno	ngos.h	err	None	Sets the global error code.
ngOSSleep	ngos.h	nticks	None	Delays task execution.
ngOSTaskCreate	ngos.h	th, task, data, prio, stack_ptr, stack_size	NG_EOK	Creates a new task.
ngOSTaskDelete	ngos.h	th	NG_EOK	Deletes a task.
ngOSTaskExit	ngos.h	none	This function does not return to its caller.	Exits the current task.
ngOSTaskGetSize	ngos.h	none	The size of the NGOStask structure	Gets the size of the task descriptor structure.
ngPollAdd	ngos.h	pent, poll_func, poll_data	NG_EOK	Adds a background routine.
ngPrintf	ngos.h	format	The number of characters written if the call is successful, otherwise a negative error Code	Writes formatted text to the standard output stream.
ngRandom	ngos.h	seed	A pseudo-random 32-bits integer	Returns a 32-bits pseudo-random number.
ngShellExit	ngos.h ngshell.h	ctx	NG_EOK	Exits the shell session.
ngShellFindCmd	ngos.h ngshell.h	ctx, cmd	A pointer to the command or NULL if the command was not found	Finds a shell command
ngShellFlush	ngos.h ngshell.h	ctx	NG_EOK	lushes the shell output stream.
ngShellGetLastRet	ngos.h ngshell.h	ctx	The return value of the last executed command	Gets the return value of the last executed shell command.

API name	Header	Parameter	Return value	Remark
ngShellInit	ngos.h ngshell.h	ctx, cmdlist, buf, cmdmax, histmax, outchar, outdata, prompt	NG_EOK	Initialises a shell session
ngShellKey	ngos.h ngshell.h	ctx, keycode	NG_EOK	Sends an input character to a shell session.
ngShellKeyAnsi	ngos.h ngshell.h	ctx, c	NG_EOK	Translates an ANSI/VT-100 character stream.
ngShellKeyPC	ngos.h ngshell.h	ctx, c	NG_EOK	Translates a PC/DOS character stream.
ngShellPutChar	ngos.h ngshell.h	ctx, c	None	Writes one character to the shell output stream.
ngShellPrintf	ngos.h ngshell.h	ctx, fmt	None	Writes formatted text to shell output stream.
ngShellPrintHelp	ngos.h ngshell.h	ctx, cmd	None	Displays help information of a shell command.
ngShellVPrintf	ngos.h ngshell.h	ctx, fmt, args	The number of characters written	Writes formatted text to the shell output stream.
ngSleep	ngos.h ngshell.h	delayms	None	Delays task execution.
ngSPrintf	ngos.h	buf, format	Length of the resulting string	Formats a string.
ngStdInChar	ngos.h	data	The next character read from the standard input or a negative value if there is no character to read	Gets the next character from standard input.
ngStdOutChar	ngos.h	c, data	None	Writes one character to standard output.
ngStrCat	ngos.h	s1, s2	A pointer to the destination string	Concatenates two strings.
ngStrChr	ngos.h	s, c	A pointer to the character or NULL if the character is not found.	Locates the first occurrence of a character in a string.
ngStrCmp	ngos.h	s1, s2	A value less than 0 if s1 is less than s2 0 if s1 is equal to s2 A value greater than zero if s1 is greater than s2	Compares two strings.
ngStrCpy	ngos.h	dst, src	A pointer to the destination string	Copies a string.
ngStrLen	ngos.h	s	The length of the string	Returns the size of a string.
ngStrNCat	ngos.h	s1, s2, n	A pointer to the destination string	Concatenates two counted strings.
ngStrNCmp	ngos.h	s1, s2, n	A value less than 0 if s1 is less than s2 0 if s1 is equal to s2 A value greater than zero if s1 is greater than s2	Compares two counted strings.

API name	Header	Parameter	Return value	Remark
ngStrNCpy	ngos.h	dst, src, n	A pointer to the destination string	Copies a counted string.
ngStrRChr	ngos.h	s, c	A pointer to the character or NULL if the character is not found	Locates the last occurrence of a character in a string.
ngStrStr	ngos.h	s1, s2	A pointer to the located string or NULL if the string was not found	Locates the first occurrence of a string inside another.
ngStrTok	ngos.h	s, sep, savs	A pointer to the token found or NULL if there are no more tokens	Breaks a string into tokens.
ngURD16, ngURD32	ngos.h	ptr	The value pointed to by ptr	Reads unaligned values.
ngURDBE16, ngURDBE32	ngos.h	ptr	The value pointed to by ptr	Reads big endian unaligned values.
ngURDLE16, ngURDLE32	ngos.h	Ptr	The value pointed to by ptr	Reads little endian unaligned values.
ngUserSetWorkProc	ngos.h	func, data	None	Sets a user background routine.
ngUWR16, ngUWR32	ngos.h	ptr, val	None	Writes unaligned values.
ngUWRBE16, ngUWRBE32	ngos.h	ptr, val	None	Writes big endian unaligned values.
ngUWRLE16, ngUWRLE32	ngos.h	ptr, val	None	Writes little endian unaligned values.
NG_VA_ARG	ngos.h	ap, type	The next argument	Gets the next argument of a variable argument list
NG_VA_END	ngos.h	ap	None	Ends a variable argument list.
NG_VA_START	ngos.h	ap, param	None	Starts a variable argument list.
ngVPrintf	ngos.h	Format, args	The number of characters written if the call was successful, otherwise a negative error Code	Writes formatted text to standard output.
ngVSPrintf	ngos.h	buf, format, args	Length of the resulting string	Writes formatted text to a string.
ngYield	ngos.h	none	None	Processes asynchronous events

CHAPTER 6

Networking

6.1 TCP/IP connection

If you want to have Ethernet access, you can adopt below methods:

- Direct connection between EVA-X1610C evaluation board and PC via an Ethernet crossover cable (Fig 6-1).
- Connection between EVA-X1610 evaluation board and PC via two Ethernet cables and a hub (Fig 6-2).

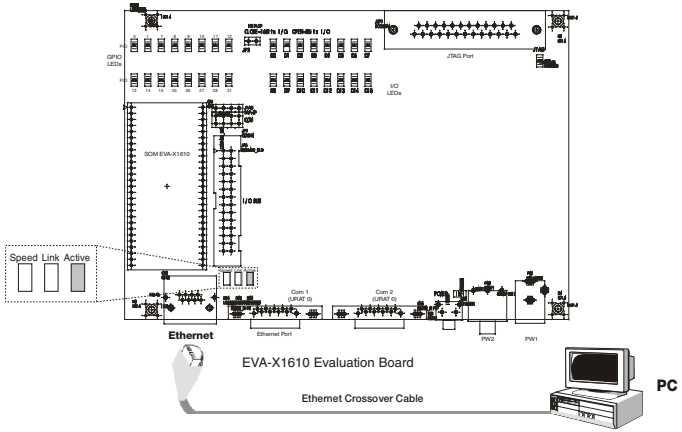


Fig 6-1: Direction connection between EVA-X1610C evaluation board and PC

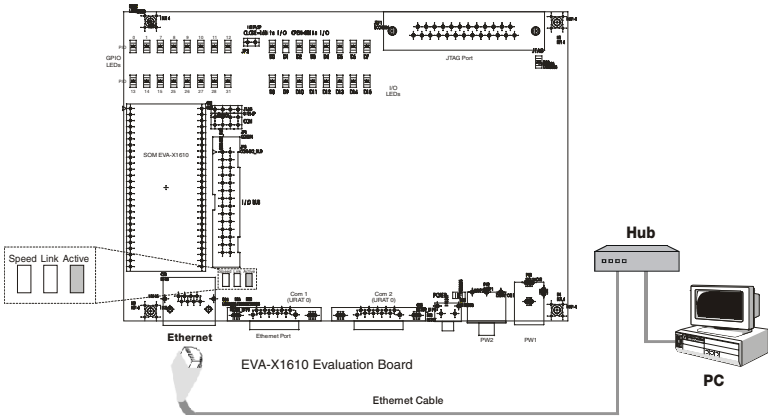


Fig 6-2: Connection between EVA-X1610C evaluation board and PC via hub

If the Ethernet connection between EVA-X1610C evaluation board and PC is built successfully, the Active LED on the evaluation board will be green.

6.2 NexGenIP

The implementation of the TCP/IP for EVA-X1610C is NexGenIP. It uses NexGenOS in order to assure CPU, complier and OS independence and use of the NexGenOS configuration mechanisms for installation of protocols and driver6.

NexGenIP includes the following protocols, installed as modules in the NexGenOS configuration table:

- IP protocol, including ICMP and IGMP
- UDP protocol, accessed through datagram sockets.
- TCP protocol, accessed through stream sockets.
- RAW-IP protocol, accessed through raw sockets.
- ARP protocol, to be used with Ethernet interfaces.

Two socket interfaces are available for communication between the application and the transport protocols: a standard BSD-like API and a proprietary NGDSock interface. Both of them are implemented on top of the asynchronous socket interface.

The following additional software can be used with NexGenIP:

Network Communication	NexGenBoot	used for dynamic configuration including the DHCP, BOOTP, and TFTP protocol.
	NexGenRESOLV	an asynchronous DNS resolver
	NexGenMAIL	client versions of the POP3 and SMTP mail protocols
	NexGenWeb	an embedded HTTP server, including a virtual file system
	NexGenPPPoE	an implementation of the PPP over Ethernet protocol, used in ADSL connection
Serial Communication	NexGenPPP	an implementation of the Point-to-Point protocol for NexGenIP

[1] Further information about NexGenOS and NexGenIP, please refer to the documentation "NexGenOS_PGuide_V1_3_20011030" and NexGenIP_Pguide_V1_3_20011031".

6.3 Configuring NexGenIP

The following introduces how to do the run-time configuration for installing and configuring the protocols and the device drivers.

In order to configure the run-time options the application must provide the NexGenOS initialization routine ngInit() with an array of NGcfgent structures called the configuration table. The configuration table contains commands for installing protocols and drivers and a list of options. Each protocol and network device has its own set of options. The option related to a protocol or an interface simply follows the protocol or interface declaration.

General Options

The general NexGenIP options must be set prior to any other commands. In order to pass the option value (or argument) the following macros are used:

Marco name	Remark
NG_CFG_INT(ival)	Integer value
NG_CFG_LNG(lval)	Long value
NG_CFG_ADR(a,b,c,d)	32-bits IP address a.b.c.d.
NG_CFG_PTR(pval)	Pointer to data
NG_CFG_FNC(fval)	Pointer to function
NG_CFG_FALSE	False (integer) value
NG_CFG_TRUE	True (integer) value

Memory allocation in the NexGenIP libraries is never dynamic. Thus, the application must reserve space for the objects that will be used by the stack. In C, this is generally done by declaring static data structures. The example below shows how to reserve space for the socket control blocks:

```
NGsock sockcb[SOCK_MAX];
```

The application must inform the stack of available resources, including message buffers used for inter-protocol communication, data control blocks used by the socket layer to store connections information, and other specific protocol related data. To install a protocol, an interface or a serial device, the corresponding command is used.

Commands	Description
NG_CFG_PROTOADD	Adds a network protocol
NG_CFG_IFADD	Adds a network interface
NG_CFG_DEVICEADD	Adds a serial device
NG_CFG_DRIVER	Specifies the driver for a network interface or a serial device. This command follows NG_CFG_IFADD or NG_CFG_DEVICEADD .
NG_CFG_END	Ends the configuration table.

Once these commands have been issued the protocol and interface options must be provided (until another section delimited with NG_CFG_PROTOADD , NG_CFG_IFADD or NG_CFG_DEVICEADD appears). The last line of the configuration table is indicated with NG_CFG_END .

The modules must be declared in the order specified below. See the description of the ngInit() function for a complete description of both general and protocol specific options and commands.

{
general options
...
serial devices
...
transport protocols (TCP, UDP)
...
network protocols (IP, ARP)
...
link-layer protocols (PPP)
...
network interfaces
...}

Message Buffers Options

Message buffer options must be set before protocols and drivers are installed. The application defines how many message buffers to use in the system, the size of the link-layer header part, and the size of the message data part.

Options	Description
NG_BUFO_MAX	Number of message buffers
NG_BUFO_HEADER_SIZE	Size of link-layer header
NG_BUFO_DATA_SIZE	Size of link-layer data (MTU)
NG_BUFO_ALLOC_F	Message buffer pool allocation routine
NG_BUFO_ALLOC_BUF_F	Message buffer allocation routine
NG_BUFO_INPQ_MAXq	Size of global input message queue

Socket Options

The following options are used to declare the socket control blocks.

Options	Description
NG_SOCKO_MAX	Number of socket control blocks
NG_SOCKO_TABLE	Pointer to socket control block table
NG_SOCKO_SEMTABLE	Pointer to socket semaphore table (RTOS mode only

RTOS Options

For a multi-task version of NexGenIP the following options must be set:

Options	Description
NG_RTO_INPUT_PRIO	Priority of the input processing task
NG_RTO_INPUT_STACK_SIZE	Size of stack
NG_RTO_INPUT_STACK	Pointer to stack
NG_RTO_TIMER_PRIO	Priority of the timer processing task
NG_RTO_TIMER_STACK_SIZE	Size of stack
NG_RTO_TIMER_STACK	Pointer to stack

In the following example the application declares its configuration table, my_config, for a polling version of the stack:

```
void my_buf_alloc( int size, void *addr);
/* socket control blocks */
NGsock sockcb[MAXSOCK];
NGcfgent my_config[] = {
NG_BUFO_MAX,                NG_CFG_INT( MAXBUF),
NG_BUFO_ALLOC_F,            NG_CFG_PTR( my_buf_alloc),
NG_BUFO_HEADER_SIZE,        NG_CFG_INT( sizeof( NGetherhdr)),
NG_BUFO_DATA_SIZE,          NG_CFG_INT( ETHERMTU),
NG_SOCKO_MAX,               NG_CFG_INT( sizeof( sockcb)/
 sizeof(sockcb[0])),
NG_SOCKO_TABLE,             NG_CFG_PTR( &socks),
NG_RTO_CLOCK_FREQ,          NG_CFG_INT( NG_CLOCK_FREQ),
/* protocol declarations follow... */
/* ... */
  NG_CFG_END
};
main()
{
ngInit( &my_config);
}
```

Add a protocol

To add a protocol, use the NG_CFG_PROTOADD command, with the pointer to the protocol structure as argument. To be able to add a protocol the header file, including the description of the protocol interface, must be included in the application. Once the protocol has been added, the protocol specific options can be set. The following describe the currently supported protocols and their related options.

- IP Protocol

The IP protocol module is mandatory when using UDP, TCP and RAW-IP protocols. The ICMP and IGMP protocols are integrated into the IP protocol module.

Header file: ngip.h

Protocol structure: ngProto_IP

Protocol Options	Description
NG_IPO_TTL	Default TTL value for outgoing datagrams.
NG_IPO_TOS	Default TOS value for outgoing datagrams. The transport protocols and the socket layer may override the default value set by the NG_IPO_TTL and NG_IPO_TOS options.
NG_IPO_FRAG_TIMEO	Time-out value in seconds for fragmented datagrams. IP fragments are stored in a reassembly queue. If no complete datagram is received during this period of time the fragments are discarded. The fragmentation option is only implemented if the IP fragmentation has been enabled at NexGenIP library creation.
NG_IPO_ROUTE_DEFAULT	Default route gateway address. Outgoing datagrams without resolved route are sent to the default gateway. A zero value disables the default gateway.
NG_IPO_ROUTE_MAX	Maximum number of entries in the routing table.
NG_IPO_ROUTE_TABLE	Location of the routing table buffer. Each routing entry is stored in a NGiprtent structure. The buffer passed with this option must be large enough to store NG_IPO_ROUTE_MAX structures.
NG_IPO_FORWARD	Enable or disable forwarding of IP datagrams. This option is only implemented if the IP forwarding has been enabled at NexGenIP library creation.
NG_IPO_SENDREDIRECT	Enable or disable sending redirect ICMP messages.
NG_ICMPO_MASKREPLY	Enable or disable mask request ICMP message replies.
NG_UDPO_CHECKSUM	Calculate UDP checksum for outgoing UDP datagram. This option is a UDP protocol option but is currently set in the IP module. The default value, NG_CFG_TRUE , should not be changed.

- ARP Protocol

The ARP protocol module must be installed when an Ethernet interface is used. This protocol maps the IP addresses to the Ethernet MAC addresses for the hosts that are on a local link.

Header file: ngeth.h

Protocol structure: ngProto_ARP

Protocol Options	Description
NG_ARPO_MAX	Maximum number of ARP host entries.
NG_ARPO_TABLE	Location of the ARP host table. Each host entry is stored in a NGarpent structure. The buffer passed with this option must be large enough to store NG_ARPO_MAX structures.
NG_ARPO_RETRY	Number of retries for the ARP requests.
NG_ARPO_WAIT	Time between two ARP requests.
NG_ARPO_EXPIRE	Time of expiration for a resolved ARP entry.
NG_ARPO_REJECT	Time during which a non-responding host is in reject state.

- UDP Protocol

The UDP protocol module must be installed if the application needs to create sockets of NG_SOCK_DGRAM type.

Header file: ngudp.h

Protocol structure: ngProto_UDP

Protocol Options: See the NG_UDPO_CHECKSUM option in the IP protocol options.

- TCP Protocol

The TCP protocol module must be installed if the application needs to create sockets of NG_SOCK_STREAM type.

Header file: ngtcp.h

Protocol structure: ngProto_TCP

Protocol Options	Description
NG_TCPO_TCB_MAX	Maximum number of TCP control blocks.
NG_TCPO_TCB_TABLE	Location of the TCP control block buffer. Each TCP socket needs additional information stored in an NGtcpcb structure, called TCP control block. The buffer passed with this option must be large enough to store NG_TCPO_TCB_MAX structures.

- RAW/IP Protocol

The RAW-IP protocol module must be installed if the application needs to create sockets of NG_SOCK_RAW type.

Header file: ngip.h

Protocol Structure: ngProto_RAWIP

Protocol Options: None

Add a network interface

To install a network interface, the NG_CFG_IFADD command, with a pointer to an empty interface data structure as argument, is added. A NG_CFG_DRIVER command, with the pointer to the driver description structure, must follow. The type of data structure is driver dependent. For Ethernet devices it is normally NGethifnet but this needs to be checked in the driver description. Interface specific options, like the interface IP address and network mask, interface name, board I/O base, and IRQ, can furthermore be added. The interface options are interface type and board dependent. The NG_CFG_IFADD command automatically opens the interface. If the interface cannot be opened at start-up (which is the case for PPP interfaces for example) the NG_CFG_IFADDWAIT command can be used instead. For a list of interface options, see ngIfGetOption()in the reference library.

Loopback Interface

The loopback interface allows a client and a server on the same host to communicate with each other using TCP/IP. Everything sent to the loopback interface is returned to the network input. The loopback interface is not installed by default and must consequently be included in the configuration table. To do this, the driver description variable ngNetDrv_LOOPBACK is used. The interface driver is implemented entirely in software and is provided as part of the NexGenIP library. The default interface address, 127.0.0.1, should not be changed.

6.4 API Functions of NexGenIP

API name	Header	Parameter	Return value	Remark
ngArpAddEntry	ngip.h ngnet.h ngeth.h	in_addr, ph_addr	NG_EOK	Adds a static ARP entry.
ngArpDeleteEntry	ngip.h ngnet.h ngeth.h	in_addr	NG_EOK	Deletes an ARP entry.
ngHTONL ngHTONS	ngip.h	value	The value in network-byte order representation	Converts host-byte order values.
ngIfClose	ngnet.h	netp	NG_EOK	Closes a network interface.
ngIfGetOption, ngIfSetOption	ngnet.h	netp, option, optval	NG_EOK	Gets or sets a network interface parameter.
ngIfGetPtr	ngnet.h	name	A pointer to the interface structure or NULL	Gets the pointer to a network interface structure
ngIfOpen	ngnet.h	netp	NG_EOK	Opens a network interface.
ngIfSetAddr	ngnet.h	netp, addr, subnetmask	NG_EOK	Sets the address and netmask of a network interface.
ngIfSetOption	See ngIfGetOption()			
NG_INADDR	ngip.h			Constructs an internet address.
ngInetATON	ngip.h	str, addr	0	Converts an ASCII string to an internet address
ngInetChecksum	ngip.h	buf, len	The internet checksum	Computes internet checksum.
ngInetNTOA	ngip.h	addr, buffer, buflen	0	Converts an internet address to an ASCII string.
ngNTOHL, ngNTOHS	ngip.h	The value to convert from network-byte order to host-byte order	Value	Converts a network-byte order value
ngPing	ngip.h ngping.h	addr, cfgping	A positive number if the target is alive, otherwise a negative error code	Send ICMP ECHO_REQUEST packets to hosts
ngRouteAdd	ngip.h	dest, subnet, gateway	0	Adds a static route
ngRouteDefault	ngip.h	gateway	0	Sets or disables the default route
ngRouteDelete	ngip.h	dest	0	Deletes a route.
ngSAIOAccept	ngip.h ngsockio.h	so, addr, newso	0	Accepts connection on a socket.

API name	Header	Parameter	Return value	Remark
ngSAIOBind	ngip.h ngsockio.h	so, addr	0	Binds local address to a socket.
ngSAIOBufAlloc	ngip.h ngsockio.h	so, bufp	0	Allocates a zero-copy buffer.
ngSAIOBufFree	ngip.h ngsockio.h	so, bufp	None	Releases a zero-copy buffer.
ngSAIOBufRecv	ngip.h ngsockio.h	so, flags, addr, bufp, dstaddr	0	Reads data on a socket with zero-copy interface.
ngSAIOBufSend	ngip.h ngsockio.h	so, bufp, flags, addr	0	Writes data on a socket with zero-copy interface.
ngSAIOClose	ngip.h ngsockio.h	so	0	Closes a socket.
ngSAIOConnect	ngip.h ngsockio.h	so, addr	0	Initiates a connection on a socket.
ngSAIOCreate	ngip.h ngsockio.h	so, domain, type, protocol, oflags	0	Creates a new asynchronous socket.
ngSAIOGetOption, ngSAIOSetOption	ngip.h ngsockio.h	so, level, optname, optval, optlen	0	Gets or sets an option on a socket.
ngSAIOListen	ngip.h ngsockio.h	so, backlog	0	Listens for connections on a socket.
ngSAIORecv, ngSAIORecvv	ngip.h ngsockio.h	so, buf, buflen, flags, from, dstaddr, iov, iovcnt	If successful the number of bytes received, otherwise a negative error code. Zero is returned when the peer has closed the connection.	Reads data from a socket.
ngSAIOSend, ngSAIOSendv	ngip.h ngsockio.h	so, buf, buflen, flags, to, iov, iovcnt	If successful the number of bytes sent, otherwise a negative error code.	Writes data to a socket.
ngSAIOSetCallback	ngip.h ngsockio.h	so, upcall_f, data	0	Attaches a callback to an asynchronous socket.
ngSAIOSetOption	See ngSAIOGetOption()			
ngSAIOShutdown	ngip.h ngsockio.h	so, how	0	Shuts down part of a socket connection
ngSockAccept	ngip.h ngdsock.h	so, addr, newso	0	Accepts connection on a socket.
ngSockBind	ngip.h ngdsock.h	so, addr	0	Binds local address to a socket.
ngSockBufAlloc	ngip.h ngdsock.h	so, bufp	0	Allocates a zero-copy buffer.

API name	Header	Parameter	Return value	Remark
ngSockBufFree	ngip.h ngdsock.h	so, bufp	None	Releases a zero-copy buffer.
ngSockBufRecv	ngip.h ngdsock.h	so, flags, addr, bufp	0	Reads data on a socket with zero-copy interface.
ngSockBufSend	ngip.h ngdsock.h	so, bufp, flags, addr	0	Writes data on a socket with zero-copy interface.
ngSockClose	ngip.h ngdsock.h	so	0	Closes a socket.
ngSockConnect	ngip.h ngdsock.h	so, addr	0	Initiates a connection on a socket.
ngSockCreate	ngip.h ngdsock.h	so, domain, type, protocol, oflags	0	Creates a new socket.
ngSockFdGetPtr	ngip.h ngdsock.h	fd, so	0	Gets a pointer to a socket control block from a socket descriptor.
ngSockGetNRead	ngip.h ngdsock.h	so	If successful the number of bytes in the socket reception buffer, otherwise a negative error code	Gets the number of bytes that can be read on a socket
ngSockGetOption, ngSockSetOption	ngip.h ngdsock.h	so, level, optname, optval, optlen	0	Gets or sets option on a socket.
ngSockGetState	ngip.h ngdsock.h	so	The state of the socket if the call was successful, otherwise a negative error code	Gets the state of a socket.
ngSockInChar	ngip.h ngdsock.h	so	A positive or NULL character value if the call was successful, otherwise a negative error Code	Reads one character from a socket.
ngSockListen	ngip.h ngdsock.h	so, backlog	0	Listens for connections on a socket.
ngSockOpen	ngip.h ngdsock.h	ctrlp, so	0	Creates a new socket.
ngSockOutChar	ngip.h ngdsock.h	c, so	None	Writes one character to a socket.
ngSockPrintf	ngip.h ngdsock.h	so, to, fmt, ...	The number of bytes sent if the call was successful, otherwise a negative error code	Writes formatted text to a socket.

API name	Header	Parameter	Return value	Remark
ngSockRecv, ngSockRecvv	ngip.h ngdsock.h	so, buf, buflen, flags, from, iov, iovcnt	Number of received bytes if the call was successful, otherwise a negative error code. Zero is returned when the peer has closed the connection.	Reads data from a socket.
ngSockSelect	ngip.h ngdsock.h	solst, nso, timeout	The number of ready sockets if the call is successful and a negative value if an error has occurred. Zero is returned when the timeout has expired.	Checks for activity on a set of sockets.
ngSockSend, ngSockSendv	ngip.h ngdsock.h	so, buf, buflen, flags, to, iov, iovcnt	Number of sent bytes if the call was successful, otherwise a negative error code	Writes data to a socket.
ngSockSetCallback	ngip.h ngdsock.h	so, upcall_f, data	0	Attaches a callback to a socket.
ngSockSetOption	See ngSockGetOption()			
ngSockShutdown	ngip.h ngdsock.h	so	0	Shuts down part of a socket connection.
ngSockVPrintf	ngip.h ngdsock.h	so, to, fmt, args	The number of sent bytes if the call was successful, otherwise a negative error code	Writes formatted text to a socket
accept	ngip.h ngsocket.h	S, addr, addrlen	A new socket descriptor for the accepted connection and -1 if an error has occurred	Accepts a connection on a socket.
bind	ngip.h ngsocket.h	s, addr, addrlen	0	Binds a name to a socket.
closesocket	ngip.h ngsocket.h	s	0	Closes a socket descriptor.
connect	ngip.h ngsocket.h	s, addr, addrlen	0	Initiates a connection on a socket.
fcntlsocket	ngip.h ngsocket.h	s, cmd,	The command result if the call is successful and -1 if an error has occurred	Manipulates a socket descriptor.
getpeername	ngip.h ngsocket.h	s, name, namelen	0	Gets name of connected peer.

API name	Header	Parameter	Return value	Remark
getsockname	ngip.h ngsocket.h	s, name, namelen	0	Gets socket name.
getsockopt, setsockopt	ngip.h ngsocket.h	s, name, optname, optval, optlen	0	Gets or sets option on a socket.
htonl, htons	ngip.h ngsocket.h	value	The converted value in network-byte order	Converts a host-byte value.
inet_addr	ngip.h ngsocket.h	str, addr	returns the internet address if the call is successful and INADDR_NONE if an error occurs	Converts a string into an internet address.
inet_aton	ngip.h ngsocket.h	str, addr	returns 1 if the string is valid and 0 if the string is invalid	Converts a string into an internet address.
inet_ntoa	ngip.h ngsocket.h	addr	A string containing the internet address in standard dot notation.	Converts an internet addresses into a string.
Ioctlsocket	ngip.h ngsocket.h	s, cmd, arg	0	Manipulates a socket descriptor.
listen	ngip.h ngsocket.h	s, backlog	0	Listens for connections on a socket.
ntohl, ntohs	ngip.h ngsocket.h	value	The converted value in host-byte order	Converts a network-byte value.
readsocket, readvsocket	ngip.h ngsocket.h	s, buf, buflen, iov, iovcnt	The number of read bytes if the call is successful and -1 if an error has occurred. Zero is returned when the peer has closed the connection.	Reads data from a socket.
recv, recvfrom	ngip.h ngsocket.h	s, buf, buflen, flags, addr, addrlen	The number of read bytes if the call is successful and -1 if an error has occurred. Zero is returned when the peer has closed the connection.	Receives a message from a socket.
recvmsg	ngip.h ngsocket.h	s, msg, flags	The number of read bytes if the call is successful and -1 if an error has occurred. Zero is returned when the peer has closed the connection.	Receives a message from a socket.

API name	Header	Parameter	Return value	Remark
selectsocket	ngip.h ngsocket.h	fdmax, readfds, writefds, exceptfds, timeout	The number of ready descriptors if the call is successful and a negative value if an error has occurred. Zero is returned when the timeout has expired.	Checks for activity on a set of sockets.
send, sendto	ngip.h ngsocket.h	s, buf, buflen, flags, addr, addrlen	The number of sent bytes if the call is successful and -1 if an error has occurred.	Sends a message to a socket.
sendmsg	ngip.h ngsocket.h	s, msg, flags	The number of sent bytes if the call is successful and -1 if an error has occurred.	Sends a message to a socket.
setsockopt	See getsockopt() .			
shutdown	ngip.h ngsocket.h	s, how	0	Shuts down part of a connection.
socket	ngip.h ngsocket.h	domain, type, protocol	A new socket descriptor if the call is successful and -1 if an error has occurred	Creates a new socket.
writesocket, writevsocket	ngip.h ngsocket.h	s, buf, buflen, iov, iovcnt	The number of sent bytes if the call is successful and -1 if an error has occurred.	Writes data to a socket.

6.5 TCP/IP sample program

We provide several sample programs to demonstrate use of TCP/IP for EVA-X1610C. Before you test those programs, you must connect PC and EVA-X1610C evaluation board together on the same network (refer to session 4.1).

The $(ngip)\tests\ucosii\i186\eva\iptest directory contains makefile for generating test program on EVA-X1610C. In a typical test configuration a terminal is connected to a serial port of the target and is used as front-end for the embedded shell included in the test program. If the target does not provide a serial line for the shell the output can be redirected to the debugger or be disabled. In the latter case a telnet client can be used to connect to the target. On most targets the debugger is used to download the application. For some targets a makefile that generates a flash image is provided.

The host application is a test server located in **$(NGIP)\tools\servtest\sertest.exe.** The test server can be compiled for a Linux or Windows host. In order to make TCP data transfer tests the target application connects to the test server. Windows test programs have been built and such that execution file also included in the shipped CD-ROM.

Test Program: testip

Makefiles dictionary: C:\EVA\adveva\i186\apps\samples\iptest\

Purpose: 1. Demonstrate the loop-back test.

2. Measure the network status and performance between EVA-X1610C (client) and PC (server).

NOTE:

(1)if you want to do the loop-back test, you must execute the test server program in your host. This program will listen network continuously. When you do the loop-back test in the client (EVA-X1610C), host will receive packets that were being sent from the client and send them back.

(2)Futher information about IP address setting, please refer to session 2.2.

The test program includes the shell commands presented below. Some commands are shipped with the standard libraries, whereas others are specific to the test program.

Standard Commands

Command	Description
Help	Displays list of commands and help information
Netstat	Displays information about the network drivers and protocols
Ifconfig	Displays information and configures network interfaces

Test Commands

Command	Description
Ping	Sends ICMP echo requests to a host. This command can be used to verify the IP communication between two hosts.
tloop	TCP loopback test. This command tries to connect to a TCP echo server and sends and receives data.
Tsend	TCP transmission test. This command tries to connect to a TCP server and sends bulk data.
Trecv	TCP reception test. This command tries to connect to a TCP server and reads bulk data.
uloop	UDP loopback test. This command tries to connect to a TCP echo server and sends and receives data.

[Example] TCP loopback test

execute the command “tloop”, it will do the TCP loopback test between client (EVA-X1610C) and server.

In this case, client sends 1024 TCP packets to server, and elapses 4460 milli-seconds. The data rate is 448 KB/sec.

```
>tloop
** TCP LOOP/BSD TEST **
msglen=1024
maxbuf=1000
1024000 bytes transfered
892 ticks elapsed
4460 milli-seconds elapsed
448 KB/s
```

6.6 TCP/IP services

6.6.1 NexGenBOOT

NexGenBOOT includes both client and server side for DHCP, BOOTP and TFTP whose descriptions are shown as below:

- **BOOTP Client.**

The BOOTP is typically used by bootstrapping systems that do not know their IP address. It can be seen as an alternative to RARP, however it allows retrieval of much more information than RARP does. As shown in Figure 6-3, BOOTP is based on an exchange of queries and answers between clients and server.

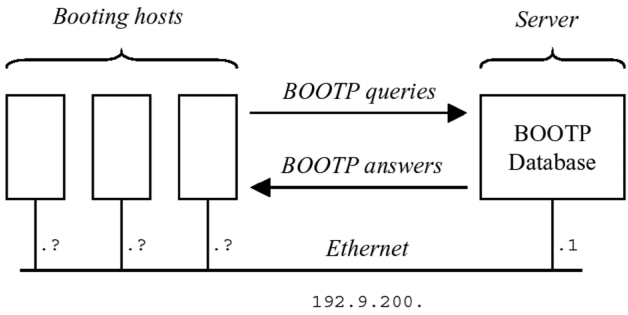


Fig 6-3: BOOTP functioning scheme

Fig 6-4 shows the overview of NexGenBOOTP client API. This package contains a single function which allows the query of a BOOTP server, i.e. ngBootpQuery().Any result is parsed internally and is returned to the application using two predefined structure types: NGbootinfo and NG-bootOptions,

Before using the ngBootpQuery() function the application must include the ngbootpc.h header file.

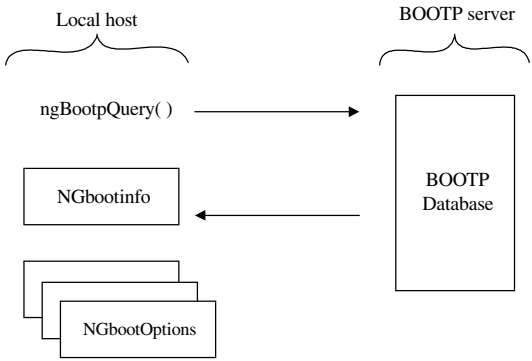


Fig 6-4: BOOTP client API overview

- **TFTP Client**

The Trivial File Transfer Protocol (TFTP) is typically used at system boot time to obtain a binary file from a server. The API presented in this chapter allows the application to perform the transfer step by step. TFTP datagrams are internally formatted, exchanged and decoded for the application.

The main goal of TFTP is to bootstrap diskless systems. The system generally uses alternative protocols, like RARP or BOOTP, to obtain an IP address before using TFTP to load a bootstrap file from a TFTP server. A TFTP client can upload a file to a TFTP server. This is however not recommended since TFTP uses the User Datagram Protocol (UDP), something that could cause security problems.

Fig 6-5 shows the overview of NexGen TFTP client API. The NexGen TFTP client package provides typical TFTP operations allowing both downloading and uploading of files from and to a TFTP server.

TFTP operations can be carried out using the following four functions: ngTftpOpen() , ngTftpClose() , ngTftpRead(), and ngTftpWrite() . Before using any of the previous functions the application must include the ngtftp.h header file.

The TFTP protocol uses the UDP as transport protocol and thus a simple configuration of the NexGen stack with an IP/UDP/Driver is necessary.

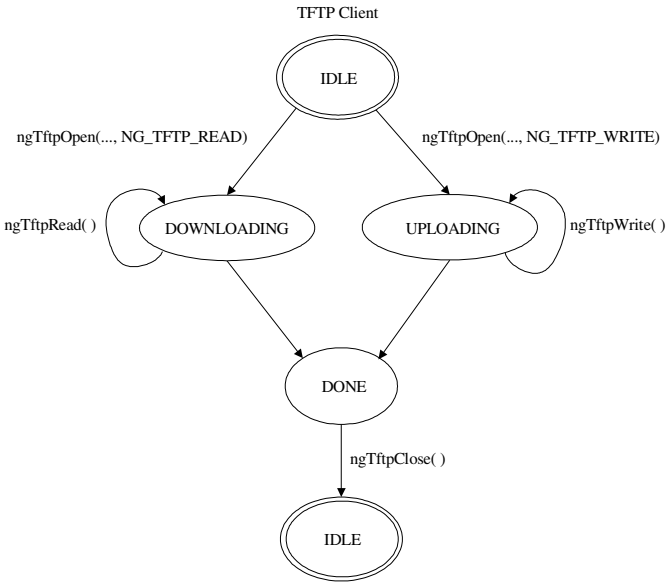


Fig 6-5 TFTP client API overview

- **DHCP Client**

The Dynamic Host Configuration Protocol (DHCP) is used to provide configuration parameters to clients. DHCP has been specifically designed to minimize the drawbacks of similar protocols, such as RARP and BOOTP. DHCP

- Reduces the management burden on the DHCP server side
- Allows dynamic host configuration (DHCP clients)
- Operates in a flexible way:
  - one server can handle many subnets
  - multiple servers are allowed
  - tolerates other BOOTP servers

The main goal of DHCP is to allocate a unique IP address to each of its clients. It can also be used to provide some configuration parameters to complete the client initialization.

Network managers should in general not have to enter any per-host configuration parameters. However, this could occur in cases where some static configurations are required.

The NexGen Software DHCP client is built as a standard NexGenIP protocol. Consequently, it can be seen as a network service daemon, which will finish the network stack configuration. However, a DHCP client requires the user to perform some basic operations by calling two functions, as shown in Figure 6-6 below. These calls are required to initiate the negotiation of a DHCP lease and to release it once the application no more wishes to use the network.

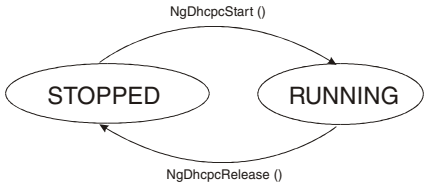


Fig 6-6: DHCP client API overview

The provided DHCP client operates in an asynchronous mode, i.e. feedback about any DHCP event is given by calling an application-defined callback. Figure 6-7 shows the interface states according to the DHCP events.

Before using any of the DHCP client functions the application must include the ngdhcpc.h header file.

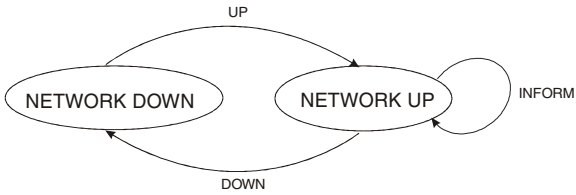


Fig 6-7 DHCP interface state (event codes)

- **DHCP Server**

In order to further contribute to the reader's understanding of the overall protocol mechanism some background information on DHCP servers will be presented. The following information is not specific to NexGen DHCP and BOOTP servers.

A DHCP server needs to:

- implement UDP as its transport protocol. DHCP messages from the server to a client are sent to the 'DHCP client' port (68)
- guarantee that no specific network address is used by more than one DHCP client at a time
- co-exist with statically configured, non-participating hosts
- retain DHCP client configuration all the way through DHCP client reboot (assigning the same configuration parameters to the client)

A DHCP server can receive one of the five following client messages:

(3) DHCPDISCOVER

(4) DHCPREQUEST

(5) DHCPDECLINE

(6) DHCPRELEASE

(7) DHCPINFORM.

The server will respond with one of the following three message types:

- DHCPOFFER - offers of some configuration parameters in response to DHCPDISCOVER
- DHCPACK - message with configuration parameters, including the granted network address
- DHCPNAK - indicates that the client's notion of the network address is incorrect

The NexGen DHCP server is built as a standard TCP/IP protocol. Consequently, it can be seen as a network service daemon, which performs basic operations for the application in order to allow the stack to answer a DHCP request. However, the user must configure the DHCP server at initialisation. The user sets the initial configuration of the server by passing the ngProto_DHCPS structure to the stack. The application can receive information or error messages from the server by setting a callback function

Before using the DHCP server API the application must include the ngdhcps.h header file.

- **BOOTP Server**

The BOOTP server can work without DHCP server, providing the user defines it in the main header file (dhcp.h) or sets ONLY_BOOTP_SUPPORTED in the makefiles. However, the BOOTP server is still part of the DHCP server and thus the user needs to define the db_globalopt option (since the BOOTP client might request some BOOTP options).

For further information about NexGenBOOT, please refer to the documentation " NexGenBOOT_pguide_V1_3_20011106".

6.6.1.1 Function APIs for NexGenBOOT

API name	Header	Parameter	Return	Remark
ngBootpQuery	ngbootpc.h	netp, binfp, retry, optList	KG_EOK	Gets boot information from a BOOTP server.
ngDhcpcRelease	ngdhcpc.h	netp	NG_EOK	Stops the DHCP client process and releases the resources on the remote server.
ngDhcpcStart	ngdhcpc.h	netp	NG_EOK	Starts a DHCP client.
ngDhcpcSetOption	ngdhcpc.h	netp, opt, optval	NG_EOK	Sets a DHCP client option on a network interface.
ngDhcpsCountstatus	ngdhcps.h	status	The number of leases that have the status defined by the status parameter	Counts the number of leases that have the status given as parameter.
ngDhcpsDebugBase	ngdhcps.h	none	None	Prints the information about each address in the DHCP server database.
NgDhcpsGetCfg	ngdhcps.h	option, arg	NG_EOK	Gets the DHCP server configuration.
ngDhcpsPrintLease	ngdhcps.h	address	NG_EOK	Prints information about a database entry.
ngDhcpsSetCfg	ngdhcps.h	option, arg	KG_EOK	Sets a DHCP server configuration.
ngDhcpsStart	ngdhcps.h	none	NG_EOK	Enables the server to start correctly.
ngDhcpsStop	ngdhcps.h	none	NG_EOK	Stops the server (allowing the application to modify the database).
ngTftpClose	ngtftp.h	tid	NG_EOK	Closes a TFTP connection.
ngTftpOpen	ngtftp.h	tid, saddr, filename, mode	NG_EOK	Opens a TFTP connection.
ngTftpRead	ngtftp.h	tid, buffer, len	Number of bytes copied into buffer (may be 0) if the call was successful, otherwise a negative error code	Reads part of a file from a TFTP server.
ngTftpWrite	ngtftp.h	tid, buffer, len	Number of bytes sent to TFTP server (should be equal to len) if the call was successful, otherwise a negative error code	Writes part of a file to a TFTP server.

6.6.2 NexGenRESOLV

NexGenRESOLV provides a so-called resolver, i.e. a local user agent, allowing applications to obtain requested host information. NexGenRESOLV has the following features:

- RTOS and polling modes supported
- Synchronous and asynchronous query calls (in both RTOS and polling modes)
- Multiple DNS servers (primary and secondary server)
- Internal cache with TTL expiration
- Configurable static host information

NexGenRESOLV is internally built as a NexGenOS protocol, which implies a good level of application transparency. As any other NexGenOS protocol, it is initialised through a configuration table (no initialisation function to call) and has its own timer (no “polling” function to call). All the application needs to do is to make sure that enough resources are available for the protocol (sufficient memory buffer for pending queries and cache entries.

Once the resolver has been successfully initialised through a configuration table, the application can call any of the resolver API functions directly.

As shown in Fig 6-8 below, the API of the DNS resolver can be devided into the following parts.

- DNS resolver service management calls
- DNS resolver query calls (ngResolv By XXX() ...) that are either synchronous calls or asynchronous calls (terminated by Async).

Before using NexGenRESOLV the application must include the ngresolv.h header file.

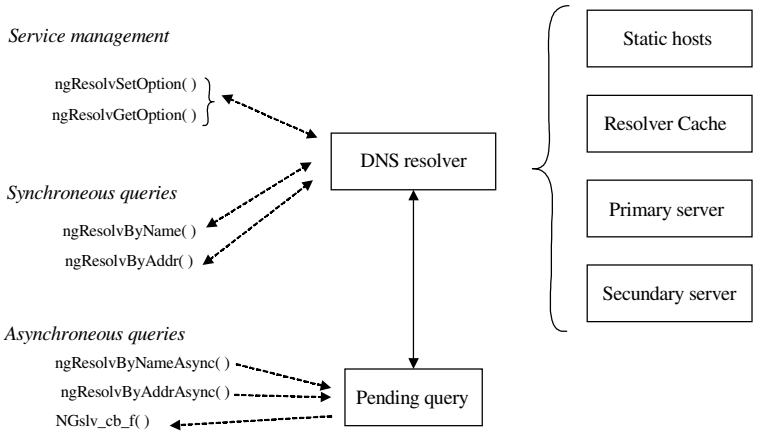


Fig 6-8 Resolver API overview

Further information about NexGenRESOLV, please refer to the documentation "NexGenRESOLV_pguide_v1_3_20011106".

6.6.2.1 Function APIs for NexGenRESOLV

API name	Header	Parameter	Return value	Remark
gethostbyaddr, gethostbyaddr_r	ngnetdb.h	addr, len, type, buf, buflen, err	A pointer to the host entry structure or NULL if the host was not found.	Gets host entry from address (BSD-compatible DNS resolver interface).
gethostbyname, gethostbyname_r	ngnetdb.h	name, result, buf, buflen, err	A pointer to the host entry or null if the host was not found.	Gets host entry from name (BSD-compatible DNS resolver interface)
ngResolvByAddr	ngresolv.h	af, addr, h, buf, buflen, flags	NG_EOK	Gets information about the host with the given IP address (blocking call).
ngResolvByAddrAsync	ngresolv.h	af, addr, h, buf, buflen, flags, cb, cbdata	NG_EOK NG_EDNS_WOULDBLOCK	Gets information about the host with the given IP address (non-blocking call).
ngResolvByName	ngresolv.h	af, name, h, buf, buflen, flags	NG_EOK	Gets information about the host with the given name (blocking call).
ngResolvByNameAsync	ngresolv.h	af, name, h, buf, buflen, flags, cb, cbdata	NG_EOK NG_EDNS_WOULDBLOCK	Gets information about the host with the given name (non-blocking call).
ngResolvGetOption	ngresolv.h	opt, optval	NG_EOK	Gets the value of a DNS resolver configuration option.
ngResolvSetOption	ngresolv.h	opt, optval	NG_EOK	Sets the value of a DNS resolver configuration option.

6.6.3 NexGenMAIL

NexGenMail includes SMTP client, POP3 client and mail parser whose descriptions are shown as below:

- **SMTP Client**

The applications can send e-mail to Internet through Simple Mail Transfer Protocol (SMTP). The SMTP defines a standard way to transfer mail between Internet hosts in a reliable and efficient manner. It is used for communication between mail capable hosts.

SMTP requires a communication channel on top of a TCP/IP stream flow. Its functioning scheme is based on a simple request/answer dialog that allows the exchange of commands, replies, and core mail messages between SMTP hosts.

Sending mail to an SMTP server requires the establishment of a TCP connection. The same connection can be used to send several messages and/or perform operations with the remote SMTP server. A connection can remain open during several mail transactions and therefore the concept of SMTP sessions is introduced.

It is possible to use either a blocking or a non-blocking mode for underlying socket operations. In case non-blocking mode is used the application must call a SMTP polling function in certain states.

Before using the NexGen SMTP client the application must include the ngsmtp.h header file.

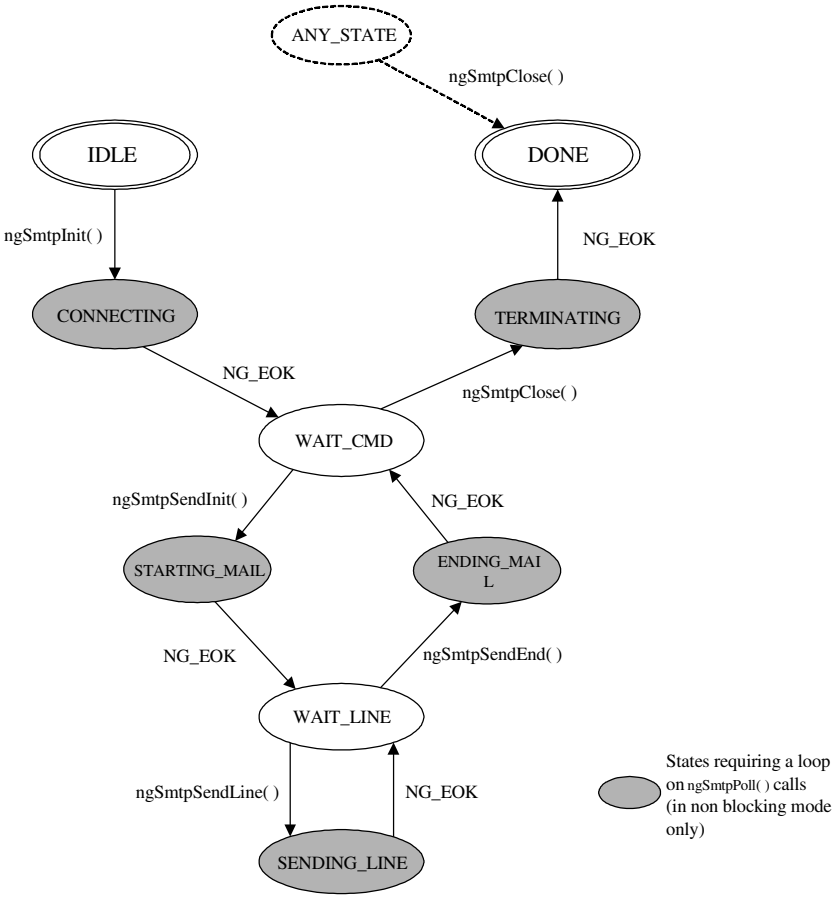


Fig 6-9: SMTP client API overview

- **POP3 Client**

NexGenMAIL POP3 client can retrieve e-mails from a POP3 server and how remote mailboxes can be managed. The Post Office Protocol version 3 (POP3) offers a simple way to download mails from a POP3 server. A POP3 server stores incoming mails for some users who have a POP3 account on that server. In this way, each registered user has a mailbox on the server side, which can be accessed after a successful authorization process.

The user can also obtain some information about his mailbox state on the server. The information is retrieved through a sequence of commands (issued by the client) and replies (returned by the server) containing the requested information. The NexGenMAIL POP3 client offers a simple set of functions based on this mechanism.

Figure 6-10 below gives an overview of the NexGenMAIL POP3 client API, including all states of a POP3 session. The POP3 session can use either blocking or non-blocking underlying I/O.

Before using the NexGenMAIL POP3 client the application must include the ngpop3.h header file.

Before using the NexGenMAIL POP3 client the application must include the ngpop3.h header file.

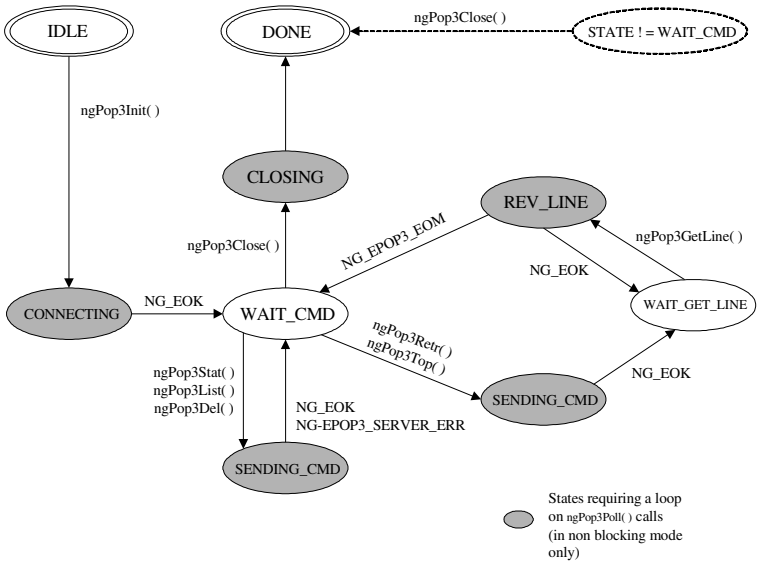


Fig 6-10: POP3 client API overview

- **Mail Parser**

The mail parsing facility of NexGenMAIL together with some background information on message formatting along with the event-driven parsing facility is presented.

The NexGenMAIL parser has the following features:

- Extensible and event-driven mail parser
- MIME version 1.0 supported (multipart messages, content-type, content-transfer-encoding, charset, etc.)
- Built-in Quoted-Printable decoding
- Base64 decoding function provided

As shown in Figure 6-11 below the API of the mail parser provided in NexGenMAIL has been divided into two distinct APIs. The mail parser takes an input byte flow through a very simple input API and sends parsed data through an event-driven output API.

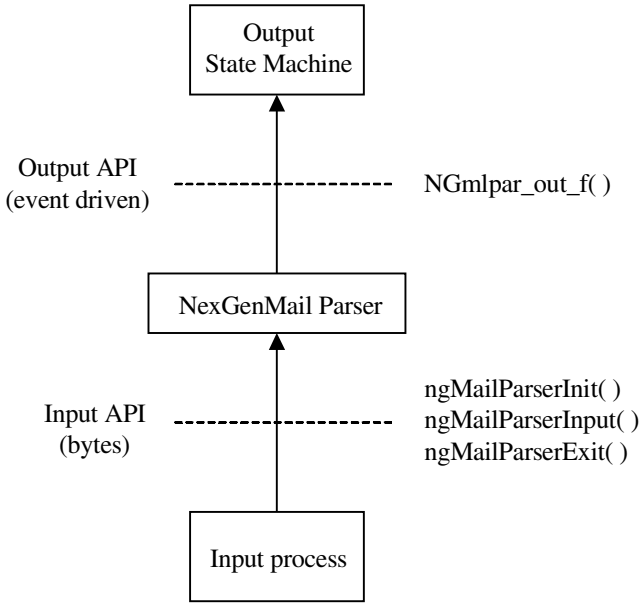


Fig 6-11: Mail parser overview

The input API is very basic, its only concern being to initialize the mail parser and to give data (e.g. the core message text) to it.

On the output side, what is done with a parsed mail is truly application dependant. It could be displayed on a screen, logged in a file, certain attachments saved to a file after decoding them, and so on. Due to the large amount of possible usages the application is left to define its own mail parsing output function.

The output API requires the application to provide an output function (prototyped by NGmlpar_out_f) in order to receive mail parser output events. These output events basically enables the identification of all parts of a message (main header, body of message, etc.) as well as the reception of raw or decoded strings from the parser. The application output function may call some external decoders in order to decode an encoded byte stream (for instance a Base64 encoded attachment could be decoded and saved in a file by the output function).

All mail parser functions are linked to the NexGenMAIL library, along with the decoding functions. Figure 6-12 gives an overview of the two APIs, summing up all aspects of the mail parser engine.

Before using the NexGen mail parser the application must include the ngmlpar.h header file.

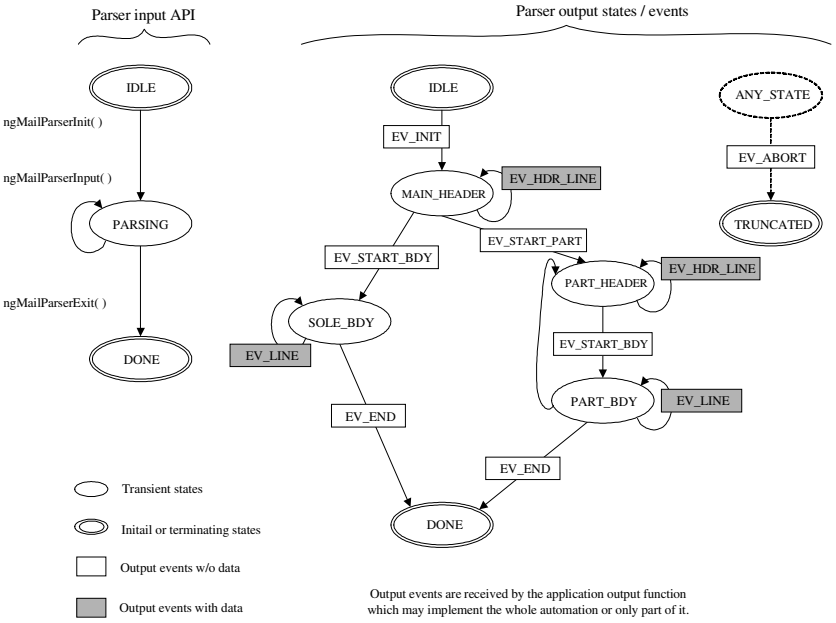


Fig 6-12: Mail parser Input/Output APIs

Further information about NexGenMAIL, please refer to the documentation “NexGenMAIL_pguide_v1_3_20011106”.

6.6.3.1 Function APIs for NexGenMAIL

API name	Header	Parameter	Return value	Remark
ngMailParserExit	ngmlpar.h	mp	NG_EOK	Terminates a mail parsing process.
ngMailParserInit	ngmlpar.h	mp, mpctl	NG_EOK	Initialises a new mail parser engine
ngMailParserInp-ut	ngmlpar.h	mp, buf, len	NG_EOKNG_EWOULDBL-OCK	Gives incoming bytes to mail parser engine.
ngPop3Close	ngpop3.h	pid	NG_EOK	Closes a POP3 connection
ngPop3Del	ngpop3.h	pid, msgId	NG_EOK	Deletes a message on the POP3 server.
ngPop3GetLine	ngpop3.h	pid, buf, len	Number of bytes written in the user buffer (length of the line) if the call was successful, otherwise a negative error code	Gets a line of a core (header or body) message from the POP3 server.
ngPop3Init	ngpop3.h	pid, args	NG_EOK	Initializes a POP3 session.
ngPop3List	ngpop3.h	pid, msgId, msgSize, numMsg	NG_EOK	Retrieves a list of message sizes.
ngPop3Retr	ngpop3.h	pid, mspId	NG_EOK	Starts retrieving process of a given message.
ngPop3Stat	ngpop3.h	pid, numMsg, mboxSize	NG_EOK	Gets statistics of the mailbox: number of messages and total size.
ngPop3Top	ngpop3.h	pid, numMsg, num	NG_EOK	Retrieves part of a message: the complete header and a number of top lines of the body text.
ngPop3Uidl	ngpop3.h	pid, numMsg, msgUidl, numMsg	NG_EOK	Gets list of unique message identifiers (UIDs).
ngSmtpClose	ngsmtp.h	sid	NG_EOK	Closes a SMTP session.
ngSmtpInit	ngsmtp.h	sid, args	NG_EOK	Initialises a SMTP client session.
ngSmtpPoll	ngsmtp.h	sid	NG_EOK G_EWOULDBLOCK	Polls an asynchronous SMTP session.
ngSmtpRecvRpl	ngsmtp.h	sid	SMTP reply code if the call was successful, otherwise a negative error code	Receives a SMTP reply in response of a ngSmtpSendCmd() call.
ngSmtpSendCmd	ngsmtp.h	sidp, cmd, arg	NG_EOK	Not available yet.
ngSmtpSendEnd	ngsmtp.h	sid	NG_EOK	Terminates the send process.
ngSmtpSendInit	ngsmtp.h	sid, cmd, from, to	NG_EOK	Initializes a mail transaction on an open SMTP session.
ngSmtpSendLine	ngsmtp.h	sid, line	NG_EOK	Sends a single line of data (core e-mail message) to a SMTP server.

6.6.4 NexGenWEB

NexGenWEB is an embedded HTTP server whose different components as well as its interactions with the application is presented in Fig 6-13 below.

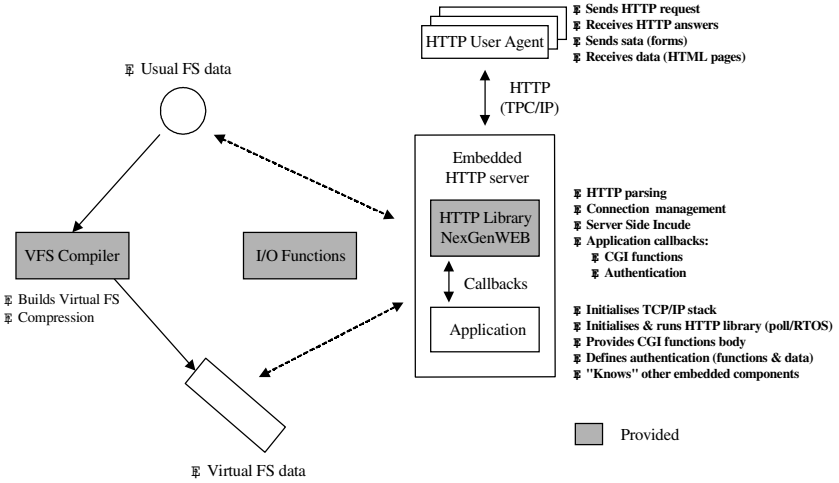


Fig 6-13: NexGenWEB components

The embedded HTTP server consists of two components: the HTTP library and the Application. The HTTP library deals with the core protocol and implements a minimal HTTP server. It manages connections (deciding whether they should remain persistent or not), takes care of parts of the resource management (associating a requested URL to the matching resource if there is one), and parses files for SSI features (on top of either VFS or UFS). Finally, this module ensures that application callbacks set at initialization are called, thus enabling CGI functions and enforcing resource and user authentication. The HTTP library only needs a small API, basically to initialise and run the HTTP server. The API changes slightly according to the underlying running mode (i.e. either polling or RTOS). The HTTP library constitutes the main part of NexGenWEB.

The second component of the embedded HTTP server is the application code. The application module in Figure 6-12 is only there to show that any application specificity can be handled. The aim of this module is to offer a possibility to glue the embedded HTTP server to other components. The application is aware of other embedded components, such as serial drivers, net cams, etc. Its code typically contains the program entry point (i.e. its main function) that should initialise a number of embedded components, a specific OS, and a TCP/IP stack before using the HTTP library API. It may also provide CGI functions and some other functions required to ensure authentication.

Two more components are important in the embedded server context and are thus briefly described: the I/O functions and the VFS compiler.

The I/O functions are used to access some underlying file system, real or virtual. Since the HTTP library uses them directly they must conform to a given high-level I/O API. Nothing prevents the application code to use them for file access for its own needs. NexGenWEB provides I/O functions for VFS and for usual files stored in some hard drive (the latter I/O functions are built on top of common C API open/close/read/...).

The VFS compiler is a tool (executable program) that imports files to the VFS format. Imported files are arranged into C source files, which are compiled and linked with the final application code. The VFS I/O functions take care of accessing VFS data at runtime. All the application needs to do is to provide the memory addresses where the data is located. This is done when initializing the HTTP library.

Building an embedded Web site is basically the same thing as building a usual Web site, i.e. to write some HTML pages. Once the pages are written they are published by running NexGenWEB.

Before actually running NexGenWEB however, the application's access to some file system needs to be checked. If the application has access to some file system, i.e. NexGenWEB is able to use the open/read/close API to access the HTML files, nothing more needs to be done. On the other hand, if the application does not have access to a file system, a Virtual File System (VFS) must be built.

The VFS compiler is used to build the VFS. What the VFS compiler does is convert the HTML and binary files into some C files to be compiled and linked with the application.

NexGenWEB provides following functions:

Common Gateway Interface Function

NexGenWEB provides a simple mechanism to allow these application defined CGI functions. As they are true callbacks, CGI functions must have a given prototype. This allows them to access usual CGI Meta variables and to benefit from the parsing already done by the HTTP library.

Authentication

As defined in RFC 2617 two kinds of authentication mechanisms can be used with HTTP:

- Basic authentication - using a Base64 encoding
- Digest authentication - using MD5 algorithm

HTTP 1.0 only supports the Basic authentication whereas HTTP 1.1 supports both. In the Basic authentication mechanism, passwords are sent without being encrypted. Although not providing a high level of security, Digest authentication is an important step in increasing the security of a web server.

Thanks to the HTTP authentication mechanism, each protected web resource can belong to a given protection space. The protection space is identified by the value of the realm parameter in the WWW-Authenticate header field. When a user agent tries to access a protected resource, the server requires the user to identify himself for that protection space. Once an authentication has succeeded, the user agent will resend the same authentication information for any resource in the same protection space.

NexGenWEB needs two application-defined functions in order to:

- define resources that require protection (and therefore user authentication)
- define users that are allowed to access a given resource in a given protection space

From the HTTP server point of view, these two functions can be seen as callbacks since the server calls them to enforce user authentication.

Server Side Include

The Server Side Include (SSI) directives are used to induce a specific behavior of the HTTP server without the user agents being aware of it. This is done by placing a specific TAG (e.g. a specific HTML entity) within the body of web published files. When the HTTP server finds such a TAG it performs the desired action. The SSI directives should be placed in files that contain a specific extension, like ".shtml" or ".sml ". NexGenWEB supports two kinds of directives: "#include" and "#exec".

There is sometimes a bit of confusion regarding the difference between CGI functions and SSI directives. Since the SSI directives are located in a HTTP published file, the SSI directives can be changed without modifications in the application code. In case a real file system is used, it does not even have to be rebuilt. When CGI functions are changed on the other hand, changes in the application code are necessary.

Further information about NexGenWEB, please refer to the documentation "NexGenWEB_pguide_v1_3_20011106".

6.6.4.1 Function APIs for NexGenWEB

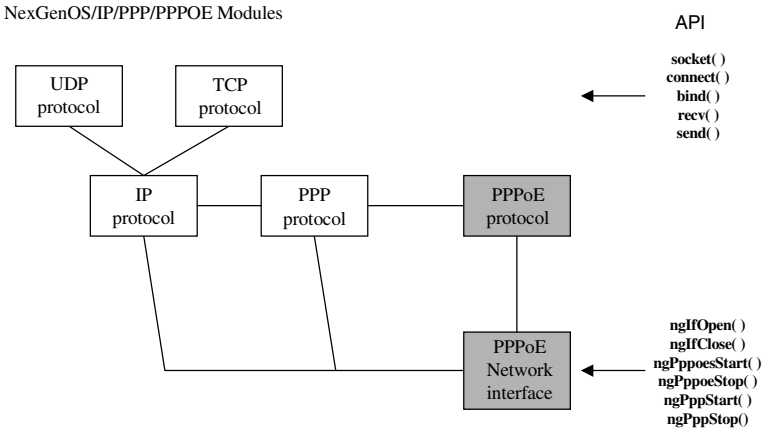
API name	Header	Parameter	Return value	Remark
ngBase64Decode	ngcodec.h	buf, len	Number of resulting bytes in the buf buffer	Base 64 decoding function.
ngBase64Encode	ngcodec.h	buf, len, tobuf, tolen	Number of bytes in the output buffer	Base 64 encoding function.
ngHttpcAbort	nghttp.h	httpc	None	This function aborts the pending request.
ngHttpcGet	nghttp.h	httpc, url, method	NG_EOK	This function gets an URL.
ngHttpcInit	nghttp.h	httpc, httpc_ctl	NG_EOK	This function initializes the Http context.
ngHttpcPoll	nghttp.h	httpc	NG_EOK	This function polls the http context.
ngHttpsExit	ngweb.h	hs	NG_EOK	Shuts down the HTTP server.
ngHttpsInit	ngweb.h	hs, hc	NG_EOK	Initialises the HTTP server.
ngHttpsPoll	ngweb.h	hs	NG_EOK	Polls HTTP server for incoming HTTP connections (polling mode only).
ngHttpsRecv	ngweb.h	wc, buf, buflen	Number of bytes written in the buf buffer or a negative error code from the socket Module	Receives data from a web connection.
ngHttpsSend	ngweb.h	wc, buf, buflen	Number of bytes sent or a negative error code from the socket module	Sends data on a given web connection.v
ngQpDecode	ngcodec.h	buf, len	Number of resulting bytes in the buf buffer	Quoted-Printable decoding function.
ngQpEncode	ngcodec.h	buf, len, tobuf, tolen	Number of bytes in the resulting buffer	Quoted-Printable encoding function.
ngUrlGetVar	ngweb.h	url, varname, buf, buflen	Number of bytes in the resulting buffer (0 means value not found)	Extracts a variable value from an URL encoded string.
ngUrlUnescape	ngweb.h	buf, buflen	Number of bytes in the resulting buf buffer.	Performs URL encoding unescaping process on a string.

6.6.5 NexGenPPPoE

NexGenPPPoE is a portable implementation of the PPPoE protocol. It uses NexGenOS in order to assure CPU, compiler, and OS independency and it uses the NexGenOS configuration mechanisms to install protocols and drivers. NexGenPPPoE must be used with NexGenPPP and NexGenIP.

The Figure 6-14 below shows the interaction between the NexGenPPPoE modules and theother NexGenOS, NexGenIP and NexGenPPP modules, a sample of the API routines that are used to access each part of the system is presented.

Fig 6-14: NexGenPPPoE modules and API



Further information about NexGenMAIL, please refer to the documentation “NexGenPPPoE_pguide_v1_3_20011226”.

6.6.5.1 Function APIs for NexGenPPPoE

API name	Header	Parameter	Return value	Remark
ngPppoeGetState	ngpppoe.h	netp	A positive value that contains the current state of the interface or a negative error code:	Gets the current internal state of a given PPPoE interface.
ngPppoeStart	ngpppoe.h	netp	NG_EOK	Starts PPPoE discovery on a given interface.
ngPppoeStop	ngpppoe.h	netp	NG_EOK	Stops the PPPoE session on a given interface.
ngPppoeWaitState	ngpppoe.h	netp, statewait, stateexit, timeoutms	NG_EOK	Waits for a given PPPoE state on an interface.

CHAPTER 7

Serial Communication

7.1 Serial communication connection

EVA-X1610C provides two serial communication ports (UART0, UART1) which are shown as Fig 7-1. For further details and communication port configuration please refer to session 3.6.

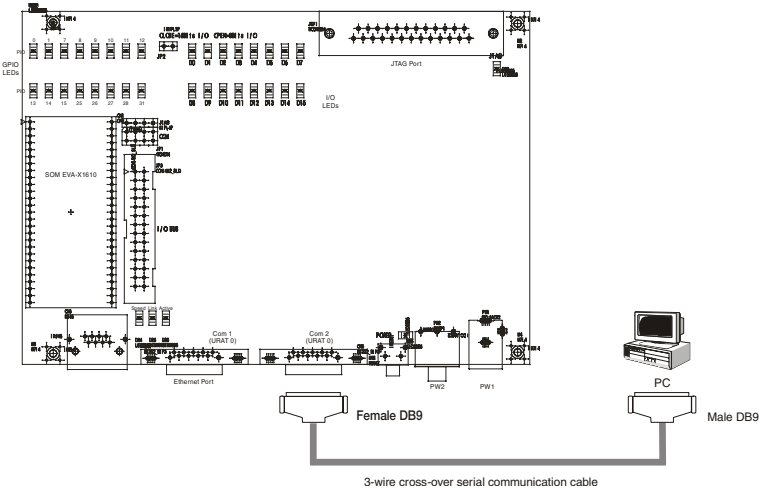


Fig 7-1: Connection between PC and EVA-X1610C Development Kit

7.2 Serial device driver

The operating system (NexGenOS) of EVA-X1610C provides a standard way of writing serial (UART) drivers. These drivers can be used directly by the application through a standard device I/O API[1] or by Peer-to-Peer Protocol (PPP)[2].

The serial device driver uses two circular buffers for receiving and sending data. In most cases an interrupt handler attached to the UART interrupt(s) handles the UART data. The driver should also provide routines to set up the link (data format, baud rate, etc.) and manage the modem lines.

The driver structure and its integration with the device I/O interface is summarized in Figure 7-2 below:

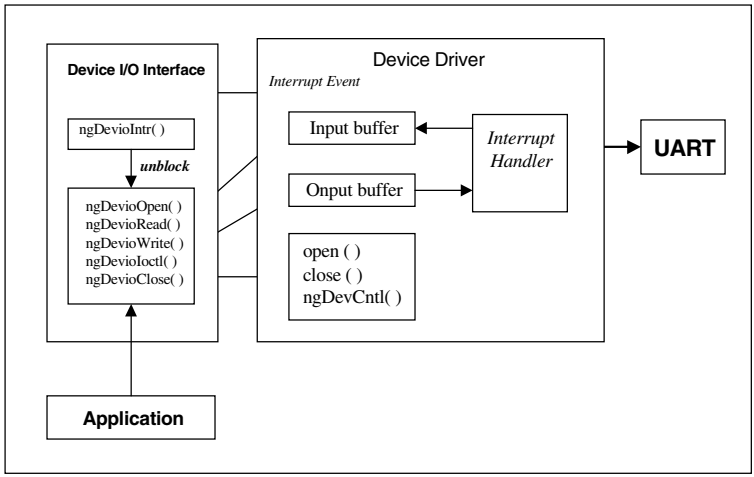


Fig 7-2: Serial device driver overview

[1] Further information about serial communication through standard device I/O, please refer to the documentation "NexGenOS_PGuide_V1_3_20011030".

[2] Further information about serial communication through Peer to Peer Protocol, please refer to the documentation NexGenPPP_PGuide_V1_3_20011203".

7.2.1 Serial Device Driver Configuration Table

The application needs to set some parameters in the NexGenOS configuration table, including:

- the creation of a pool of serial devices. An NGdevcb structure is used for the serial device control block (one for each device)
- the serial device itself. An NGdev structure is used to define the context and all relative items for a serial device
- the serial device driver. Each driver is attached to at least one logical serial device.

To be able to initialise the serial devices and their control blocks, the NexGenOS header file, ngdev.h, must be included by the application. To initialise the serial drivers, the corresponding header file of the driver must furthermore be included. All provided serial drivers are found in the $(NGOS)/drivers directory.

In the example below, the NexGenOS configuration table is shown. In order to keep the example as general as possible, everything which is driver dependent has been replaced by DEV_XXX macro.

```
#include <ngos.h>
#include <ngdev.h>
/*
 * include the serial device header file (from dir "ngos/drivers")
 * and define serial driver relevant macros DEV_XXX (see example below)
 */
/* serial device */
NGdev dev0;
NGubyte dev0_ibuf[DEV_IBUF_SIZE];
NGubyte dev0_obuf[DEV_OBUF_SIZE];

/* serial device control block pool (only 1 here) */
NGdevcb devcbs[1];

NGcfgent my_config[] = {
/* add serial device control block pool */
NG_DEVCBO_TABLE, NG_CFG_PTR( devcbs),
NG_DEVCBO_MAX, NG_CFG_INT( sizeof( devcbs)/sizeof( devcbs[0])),
```

```
/* add a serial device */
NG_CFG_DEVADD, NG_CFG_PTR( &dev0),
NG_CFG_DRIVER, NG_CFG_PTR( &DEV_DRIVER),
NG_DEVO_NAME, NG_CFG_PTR( "/dev/ser0"),
NG_DEVO_IBUF_PTR, NG_CFG_PTR( dev0_ibuf),
NG_DEVO_IBUF_LEN, NG_CFG_INT( sizeof( dev0_ibuf)),
NG_DEVO_OBUF_PTR, NG_CFG_PTR( dev0_obuf),
NG_DEVO_OBUF_LEN, NG_CFG_INT( sizeof( dev0_obuf)),
#ifdef DEV_IRQ
NG_DEVO_IRQ, NG_CFG_INT( DEV_IRQ),
#endif
#ifdef DEV_IOBASE
NG_DEVO_IOBASE, NG_CFG_INT( DEV_IOBASE),
#endif
#ifdef DEV_CLKBASE
NG_DEVO_CLKBASE, NG_CFG_INT( DEV_CLKBASE),
#endif
#ifdef DEV_SPEC1
NG_DEVO_SPEC1, NG_CFG_INT( DEV_SPEC1),
#endif
#ifdef DEV_SPEC2
NG_DEVO_SPEC2, NG_CFG_INT( DEV_SPEC2),
#endif
#ifdef DEV_DATA
NG_DEVO_DATA, NG_CFG_INT( DEV_DATA),
#endif
#ifdef DEV_CFLAGS
NG_DEVO_CFLAGS, NG_CFG_INT( DEV_CFLAGS),
#endif
#ifdef DEV_IFLAGS
NG_DEVO_IFLAGS, NG_CFG_INT( DEV_IFLAGS),
#endif
#ifdef DEV_OFLAGS
NG_DEVO_OFLAGS, NG_CFG_INT( DEV_OFLAGS),
#endif
#ifdef DEV_BAUDS
NG_DEVO_BAUDS, NG_CFG_INT( DEV_BAUDS),
#endif
/* protocols declarations follow... */
/* ... */
NG_CFG_END
};
main()
```

```
{
if (NG_EOK != ngInit( &my_config)) {
ngExit( 1);
}
/* serial device API calls (see ngDevioXXX) */
ngExit( 0);
}
```

The following example shows a serial driver installation using the 8250 serial driver to open port COM2 at speed 115200 on a PC.

```
#include <drivers/i8250.h>
#define DEV_DRIVER ngSerDrv_I8250
/* use COM2 */
#define DEV_IRQ 3
#define DEV_IOBASE 0x2f8
/* 115200, 8 data bits, 1 stop bit, no parity, no flow control */
#define DEV_BAUDS 115200
#define DEV_IFLAGS 0
#define DEV_OFLAGS 0
/* buffers size */
#define DEV_IBUF_SIZE 256
#define DEV_OBUF_SIZE 256
```

7.2.2 Serial Device API

The NexGenOS serial device Application Programming Interface (API) is defined by the ngdev.h header file.

The API defines a set of functions that work with a logical serial device. The following three serves can be offered:

- opening/closing a serial device
- reading from/writing to a serial device
- setting/getting I/O management.

Further details about serial device API are offered in the documentation “NexGenOS_Pguide_v1_3_20011030”.

Opening and closing a serial device

Before it is possible to write to or to read from a serial device the underlying device must be opened using the following call:

```
int ngDevioOpen( NGdev *devp, int flags, NGdevcb **dcb);
```

> This function creates an I/O control block associated with a serial device that will be used to perform basic I/O operations. The device is always opened in an exclusive way.

When no more data is to be sent or received, the connection should be closed, using the following call:

```
int ngDevioClose( NGdevcb *dcb);
```

> This function closes a device previously opened with ngDevioOpen().

The following example shows how to open and close a serial device:

```
#include <ngdev.h>
/* serial device */
NGdev dev0;
NGcfgent my_config[] = {

  /* see previous example of configuration table */
NG_CFG_END
};
main()
{
NGdevcb *dcb;       /* local handle to serial device control block */
if (NG_EOK != ngInit( &my_config)) {
ngExit( 1);
}
/* open serial device */
if (NG_EOK != ngDevioOpen( &dev0, 0, &dcb)) {
ngExit( 1);
}
/* read/write serial device API calls (as described in sections 3.3.2 and
3.3.3) */
/* close serial device */
ngDevioClose( dcb);
ngExit( 0);
}
```

Reading from serial devices

The API provides two basic functions to read data from serial devices:

```
int ngDevioRead( NGdevcb *dcb, void *buf, int buflen, int flags);
```

```
int ngDevioReadByte( NGdevcb *dcb, int flags);
```

The first function reads an amount of data whereas the second one is used when the data flow is character oriented. Both functions take flags as argument, used to set the kind of the read operation. It can take the following values:

NG_IO_NONBLOCK	The read operation is used in non-blocking mode
NG_IO_WAITALL	The call should block until all data is received

It is also possible to issue a read operation with timeout using the following function:

```
int ngDevioReadEx( NGdevcb *dcb, void *buf, int buflen, int min,
```

```
u_long timeo);
```

The min argument states the minimum number of bytes required before the call returns and timeo is used to set the reading timeout.

Writing to serial devices

To write data through a serial device, the following function is normally used:

```
int ngDevioWrite( NGdevcb *dcb, void *buf, int buflen, int flags);
```

The flags argument is used to set the mode of the write operation and it can take the following values:

NG_IO_NONBLOCK	The write operation is used in non-blocking mode
NG_IO_WAITALL	The call should block until all data has been sent

When the data to be written is not a binary stream, the two following functions can be used:

```
int ngDevioPrintf( NGdevcb *dcb, const char *fmt, ...);
```

```
int ngDevioVPrintf( NGdevcb *dcb, const char *fmt, NGva_list args);
```

These calls are used like ngPrintf(). For further information on formatting capabilities, see ngPrintf().

I/O Management

In order to set or get an option on a device, the following call is used:

```
int ngDevioIoctl( NGdevcb *dcb, int level, int cmd, void *arg, int
*arglen);
```

Redirecting standard I/O

NexGenOS provides a means to redirect standard input (stdin) and standard output (stdout) to a serial port. The two functions below are used to write and read data, respectively, on a serial device. The data argument must be a pointer to a serial device control block (NGdevcb) structure.

```
void ngDevioOutChar( int c, void *data);
```

```
int ngDevioInChar( void *data);
```

7.3 Peer to Peer Protocol NexGenPPP

In Figure 6-3 below, PPP's and other protocols' position and interaction in the network layers are presented. As shown, PPP is located in the Link layer between the Network and the Physical layers. The PPP used with an asynchronous line consists of four distinct components: the asynchronous HDLC (AHDLC), the Link Control Protocol (LCP), the authentication protocols (AUTH), and the IP Control Protocol (IPCP).

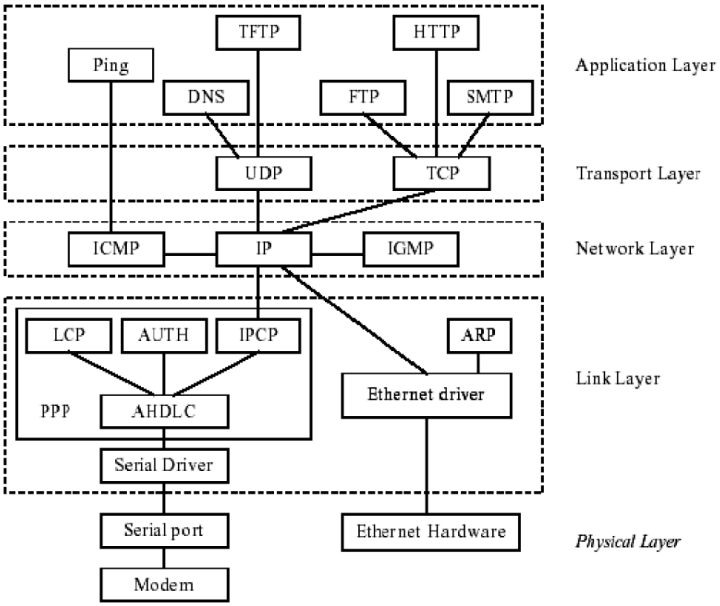


Fig 7-3: PPP in the network layers

NexGenPPP is a portable implementation of the PPP suite of protocols. It uses NexGenOS in order to assure CPU, compiler, and OS independency and it uses the NexGenOS configuration mechanisms to install protocols and drivers. The AHDLC protocol is built on top of the serial driver interface defined by NexGenOS. NexGenPPP must be used with NexGenIP.

As PPP is often used to connect to the Internet via some Internet Service Provider (ISP), NexGenPPP also offers means to manage modems in an easy way. NexGenPPP lets the application define modem chat scripts in order to make any kind of modem manipulation possible.

The Figure 7-4 below shows the interaction between the NexGenPPP modules and the other NexGenOS and NexGenIP modules, a sample of the API routines that are used to access each part of the system is presented. The installation and configuration of the PPP modules and the usage of the PPP routines is described in the next chapters.

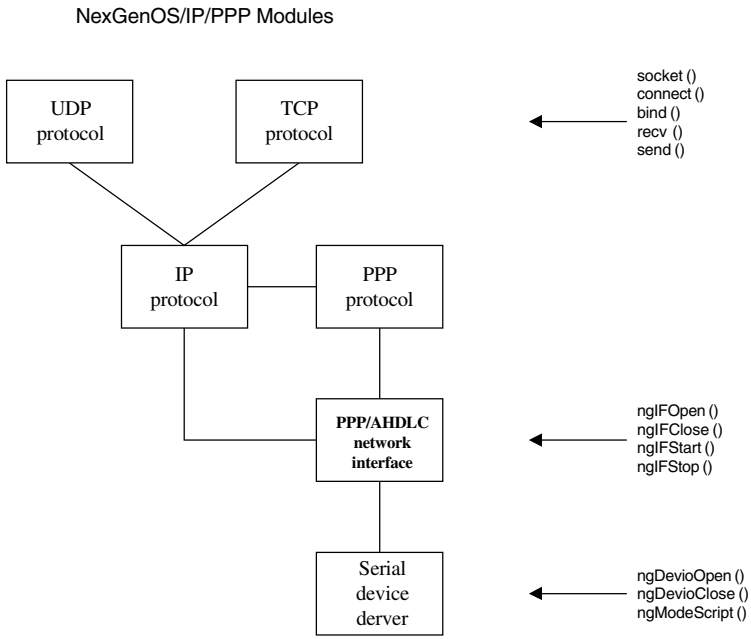


Fig 7-4: NexGenPPP modules and API

7.3.1 NexGenPPP API

API name	Header	Parameter	Return	Remark
ngModemDropLi-nes	ngdev.h ngmdm.h	diop, droptime, timeout	NG_EOK	Initialises modem lines (hardware modem reset).
ngModemInit	ngdev.h ngmdm.h	mdm, diop, mdmscript	NG_EOK	Initialises a script execution (polling interface).
ngModemPoll	ngmdm.h	mdm	A positive user defined code if the script has ended or a negative error code if the script has failed.	Polling function for modem script execution (polling interface).
ngModemScript	ngmdm.h	diop, mdmscript	A positive user defined code if the script has ended or a negative error code if the script has failed.	Launches modem script interpreter.
ngPppGetState	ngppp.h	netp	A positive value that contains the current state of the interface or a negative error code:	Gets the current internal state of a given PPP interface.
ngPppStart	ngppp.h	netp	NG_EOK	Starts PPP negotiation on a given interface.
ngPppStop	ngppp.h	netp	NG_EOK	Stops the PPP interface.
ngPppWaitState	ngppp.h	netp, statewait, stateexit, timeoutms	NG_EOK	Waits for a given PPP state on an interface.

7.4 Running a Serial Communication Sample Program

Make file directionary: **$(ADVEVA)/i186/apps/samples/pppcli (Client; dial up)**

$(ADVEVA)/i186/apps/samples/pppserv (Server; dial in)

Purpose: peer-to-peer serial communication via a modem or a direct connection.

The PPP test programs are based on a unique source code, ppptest.c, which can be configured as a PPP server or a PPP client, using a modem or a direct connection. The $(ADVEVA)/i186/apps/samples/pppserv directory contains the server version and the $(ADVEVA)/i186/apps/samples/pppcli directory contains the client version of the program. A typical configuration for the PPP server program is given in Figure 7-5 below. Two serial lines are used: one for the PPP connection and the other as a front-end for the embedded shell included in the test program. If the target does not provide a second serial line for the shell the output can be redirected to the debugger or be disabled. In the latter case the connection to the target can be done with a telnet client once the PPP connection is established.

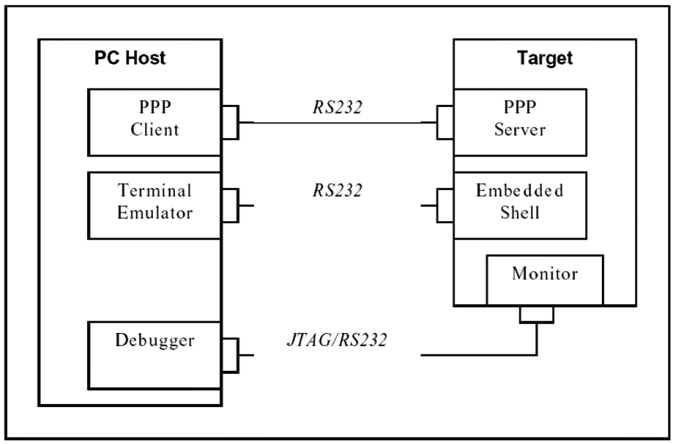


Fig 7-5: Host-target connection

When testing the PPP client program either a modem or a direct connection can be used to connect to the target. The shell is however mandatory for the client program since the connection is started by a command. The test program can be modified so that it automatically starts a connection.

On most targets the debugger is used to download the application. With some targets a makefile that generates a flash image is provided.

NOTE: if you want to do the loop-back test, you must execute the test server program in your host. This program will listen network continuously. When you do the loop-back test in the client (EVA-C1610), host will receive packets that were being sent from the client and send back.

7.4.1 Shell Command

Standard Commands

Command	Description
Help	Displays list of commands and help information
Netstat	Displays information about the network drivers and protocols
Ifconfig	Displays information and configures network interfaces
devstat	displays information about the serial drivers

Client/Server Version Commands

Command	Description
Info	displays connection information
Login	sets the PPP user name
Passwd	sets the PPP password
Stop	closes the PPP link

Client Version Commands

Command	Description
Phone	sets the phone number to call
Connect	starts the modem script and connects to the server

Test Commands

Command	Description
Ping	Sends ICMP echo requests to a host. This command can be used to verify the IP communication between two hosts.
tloop	TCP loopback test. This command tries to connect to a TCP echo server and if that succeeds it sends some data. The servtest program that is shipped with NexGenIP can be used on the host.

7.4.2 Configuring the Target

The test program includes the ngtest.h header file that contains some configuration options. Some options can also be configured in the make-file. The options set in the header file are listed below:

PPP Options

Options	Description
WANT_PPP_SERVER	configures the program as a server. If this option is not set the program is configured as a client.
WANT_DEBUG	enables debug traces
WANT_MODEM	enables modem configuration. If this option is not set a direct connection configuration is used.
WANT_VJCOMP	enables Van Jacobson compression
WANT_PAP	enables PAP authentication
WANT_CHAP	enables CHAP authentication

IP/Network Options

Options	Description
BUF_MAX	total number of message buffers
BUF_HDRMAX	size of link layer header (PPP header = 4)
BUF_DATAMAX	size of link layer data (typically 1500 + 2 bytes for the PPP checksum)
BUF_POOLSIZE	size of the message buffer pool
SOCK_MAX	maximum number of open sockets

Serial Driver Options

Options	Description
DEV_DRIVER	name of the driver (the driver object code must be declared in the makefile)
DEVDRVDATA	type of device data structure. NGdev by default.
DEV_BAUDS	baud rate
DEV_IBUF_SIZE	size of input buffer
DEV_OBUF_SIZE	size of output buffer

RTOS Specific System Options

Options	Description
STACK_SIZE	size of stack
DEV_PRIO	priority of device management task
TIMER_PRIO	priority of timer task
INPUT_PRIO	priority of network input task

Standard Input/Output Options

Options	Description
STDIO_INIT	standard input/output initialisation
SHELL_KEY_FILTER	input data processing filter

Application Options

Options	Description
CONN_MAX	number of TCP echo servers
CONN_PORT	port number of TCP servers

RTOS Specific Application Options

Options	Description
APP_DEVTASK_PRIO	priority of serial line management task
APP_ECHTASK_PRIO	priority of TCP echo server task
APP_TELTASK_PRIO	priority of telnet server task

7.4.3 Configuring the Host

Despite the symmetric aspect of PPP, configuring a PPP "client" normally means that:

- a valid IP address is requested from the other point
- some authenticating information is sent
- the other point is not forced to authenticate itself

7.4.4 Windows 2000 Direct Connection

Before installing the connection, a null-modem device must be installed by the administrator as described below:

- Go to menu: Start/Settings/Control Panel/Phone and Modem Options.
- Select Modems.
- Add a new device: "Communication cable between two computers".

When this is done any standard user can install the direct connection, using the following description:

- Go to menu: Start/Settings/Network and Dial-up Connections/Make New connection
- Select option: "Connect directly to another computer".
- Select device: "Communication cable between two computers".

A dialog box will open. Go to Properties and then configure the device with the right baud rate in the General tab. Enable the hardware flow control if it has been enabled on the target.

In order to speed-up the connection time, disable the protocols other than TCP/IP in the Networking tab. Select the authentication protocol in the Security tab. NexGenPPP supports PAP and standard CHAP. Microsoft CHAP is however not supported. However, if there is no need to speed up the connection the Security option can be set to its default value (Typical).

At this point, type the user name (default is pppuser) and password (default is 123456) and press Connect.

The Windows 2000 PPP client sends the “CLIENT” string and waits for the “CLIENTSERVER” string before starting PPP. The serial management task in the test program looks for the T character, replies with the “CLIENTSERVER” string and then starts PPP.

[Example] Make the pear-to-pear connection

<Step1>download ppp client program (the make file is in $(ADVEVA)/i186/apps/samples/pppcli) to the firmware of EVA-X1610C development board.

<Step2>connect EVA-X1610C evaluation board to modem.

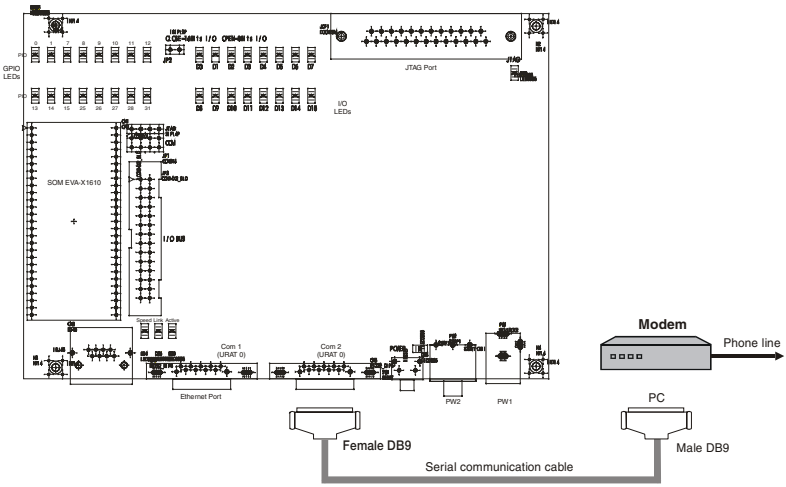


Fig 7-6: The connection between EVA-X1610C evaluation board and modem

<Step3> Turn EVA-X1610C evaluation board on and it will show below message.

```
>NexGenPPP Client Demo started
NexGenOS v1.3b, Copyright (c) 2002 WWW.NexGen-Software.fr
Wait for telnet connection at port 23
Wait for connections at port 6000
Wait for connections at port 6001
Available commands:
  phone ####        set the phone number
  login username    set the user name
  passwd xxx        set the password
  connect           connect to ISP
  stop              stop the connection
  help              display help and list of commands
```

<Step4> Enter the dial-up phone number, user id and password.

```
>phone 1234567
>login user1
>passwd 111
```

<Step5> make the pear-to-pear connection.

```
>connect
reset modem...
Starting modem script:
modem: OK
starting PPP
PPP-CB: link:  UP -> ESTABLISH ()
>PPP-CB: link:  OPEN -> AUTH ()
PPP-CB: link:  PAP : Local successfully authenticated
PPP-CB: link:  SUCCESS -> NETWORK ()
PPP-CB: link:  IPCP UP -> NETWORK ()
PPP-CB: link:  IPCP CONFIGURED -> NETWORK ()
PPP-CB: Link connected, local=163.30.208.231, dest=168.95.77.45
```

<Step6> You can use the command “ping” to confirm the connection is built successfully.

```
>ping 202.1.237.21
32 bytes from 202.1.237.21: icmp_seq=0 ttl=249 time=141 ms
32 bytes from 202.1.237.21: icmp_seq=1 ttl=249 time=146 ms
32 bytes from 202.1.237.21: icmp_seq=2 ttl=249 time=156 ms
32 bytes from 202.1.237.21: icmp_seq=3 ttl=249 time=146 ms
```

CHAPTER 8

LCD Option

8.1 EVA-X1610C Evaluation Board with LCD Connection

If you want to connect the EVA-X1610C evaluation board with LCD (EVA-X1610C-DK1) to PC, you can refer to Fig 7.1.

NOTE:

The LCD in EVA-X1610C-DK1 supports touchscreen function. If you want to enable the Touch Screen function for an LCD panel, you can configure it by setting jumper JP3, JP4 and JP5. For further information about the configuration of JP3, JP4, JP5, please refer to session 3.8.

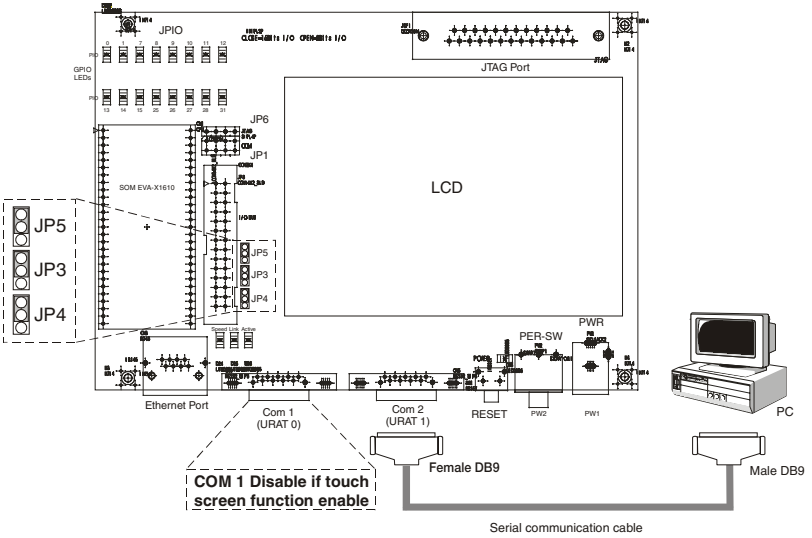


Fig 8.1: EVA-X1610C-DK1 connction

8.2 Graph Service NexGenGRAPH

The implement action of graph for EVA-X1610C is NexGenGRAPH. It has been built on top of the NexGenOS shell. Further information about NexGenOS, please refer to the NexGenOS Programming Guide.

NexGenGRAPH is a graphical user library specifically designed for embedded environments. NexGenGRAPH is designed to be used directly by applications that need a graphical interface.

NexGenGRAPH performs common graphical operations, such as plotting pixels, lines, boxes, circles, and ellipses, and drawing bitmaps and images. A font manager is also provided in NexGenGRAPH used to display text in customisable fonts.

Apart from the library, NexGenGRAPH relies on a layer called the video driver. This layer supports different screen resolution modes. It manages the graphical controller using the common functions offered by the library or the accelerated operations if the video chip supports them.

Furthermore, NexGenGRAPH supplies some input drivers such as keyboard and pointing device drivers to manage user inputs. An event manager module is also provided including a set of functions to queue user events so that they can be directly used by applications.

Fig 7-2 gives an overview of the NexGenGRAPH architecture for the graphical part.

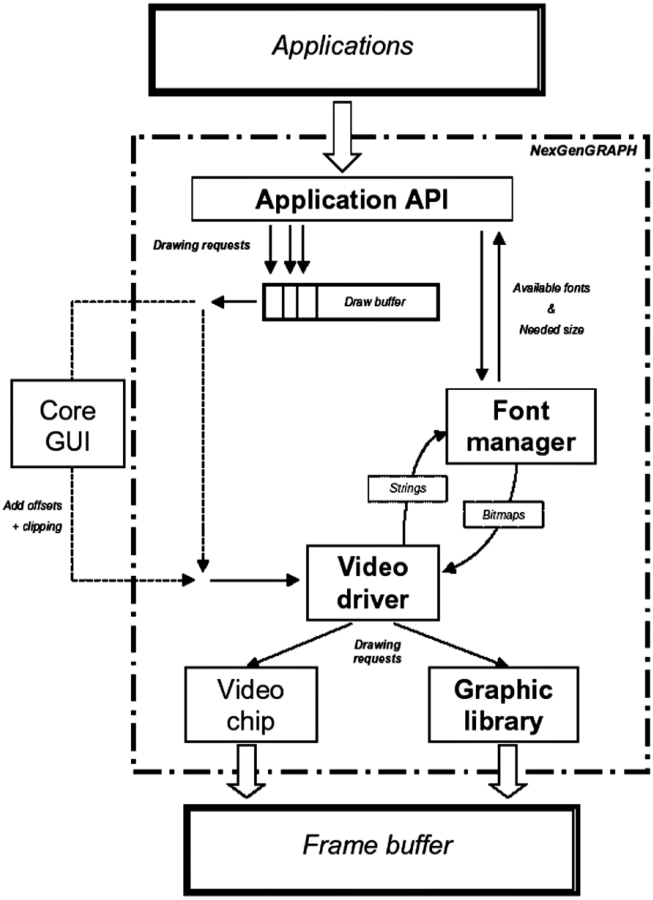


Fig 8-2 The NexGenGRAPH architecture

When writing a graphical application, consideration has to be taken to how the video controller works since it is responsible for the screen management. The video controller must perform the drawing commands sent by the application and transform them into visible objects on the screen. Thus, all drawing commands must access the VRAM (Video RAM), which is organised as a buffer, a so-called frame buffer.

In order to obtain portability, a graphical library needs to provide an Application Programming Interface (API). Using this API, the application does not need to know which video controller is used, i.e. enabling hardware abstraction. NexGenGRAPH offers this kind of API, thus enabling the application to make graphical requests. The requests are stored in a so-called draw buffer and are carried out when the draw buffer is sent to the video driver to be executed.

The request mechanism decreases the number of time-consuming accesses to the video driver, by buffering and concatenating requests of the same type. Each request is buffered with its arguments. For instance, a draw line request is associated with its start and end point coordinates and with a colour. A box request is followed by its origin, width and height parameters, drawing and filling colours, texture to apply, etc.

The video driver, in its turn, manages all accesses to the hardware video. It uses the correct set of functions supplied by NexGenGRAPH to manage the frame buffer and to display the graphical requests. It can also use specific accelerated operations if the video controller supports them. An internal font manager (FM) is used to turn text strings into bitmaps according to the selected font. The video driver thereafter displays the bitmap on the screen.

The buffer that contains the requests is flushed (executed) either:

- automatically when the buffer is full and a new request needs to be added by the application.

or

- by the application calling a specific function. This function is typically called when an application needs to refresh a part of the screen.

The following diagram shows the request mechanism used by NexGen-GRAPH:

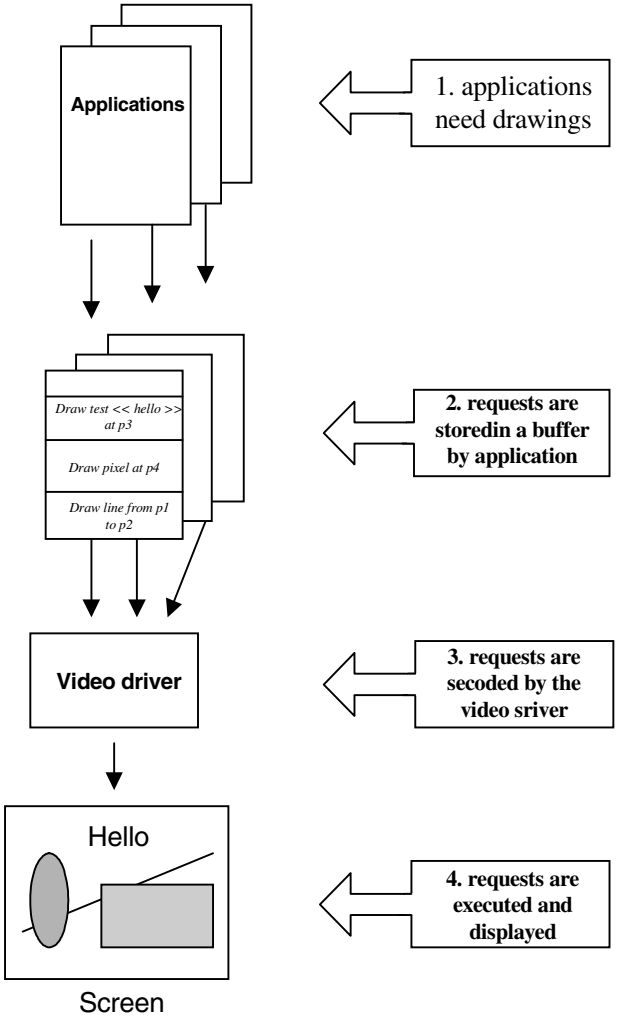


Fig 8-3: Request Mechanism

Another reason to use an API is to insure a safe RTOS functioning. Since all video driver and font manager accesses are locked when using the API functions simultaneous access is avoided.

NexGenGRAPH can work with or without an external core Graphical User Interface (GUI). By using a core GUI, such as NexGenGUI4(r) many applications can be managed on the same platform using the same screen.

For more information about NexGenGRAPH, please refer to the documentation “NexGenGRAPH Programming Guide_dev_rel_1_3_b_1”.

8.3 API Functions of NexGenGRAPH

API name	Source	Parameter	Return	Remark
nggDrawAddClip	nggraph.h	drawctx, newclip, oldclip	NG_EOK	Adds a clipping region to the current draw context.
nggDrawAddOffset	nggraph.h	drawctx, offset, oldoffset	NG_EOK	Adds an offset to the current one.
nggDrawArc	nggraph.h	drawctx, r, start, end, ss	NG_EOK	Displays an arc of a circle or an ellipse with or without filling.
nggDrawBitmap	nggraph.h	drawctx, bitmap, region, p, cp	NG_EOK	Displays a bitmap.
nggDrawBox	nggraph.h	drawctx, r, ss	NG_EOK	Draws a box.
nggDrawCircle	nggraph.h	drawctx, r, ss	NG_EOK	Displays a circle or an horizontal or vertical ellipse .
nggDrawCtxInit	nggraph.h	drawctx, drawbuf, buflen, lush_f(),flush_data, fm	NG_EOK	Initialises a draw context attached to a video driver.
nggDrawFlush	nggraph.h	drawctx	NG_EOK	Flushes the draw context buffer.
nggDrawGetDrawColor	nggraph.h	drawctx	The drawing color.	Gets the drawing color.
nggDrawGetDrawMode	nggraph.h	drawctx	The drawing code.	Gets the drawing mode.
nggDrawGetFillColor	nggraph.h	drawctx	The filling colour.	Gets the filling colour.
nggDrawGetFillPat	nggraph.h	drawctx, fillpat	NG_EOK	Gets the pattern used to fill shapes.
nggDrawGetFillPatColor	nggraph.h	drawctx	The color of the filling pattern.	Gets the color used for the filling pattern.
nggDrawGetFlags	nggraph.h	drawctx	The flag value.	Gets the draw context flag.
nggDrawGetFontCtx	nggraph.h	drawctx	The font context handle.	Gets the font context handle of a draw context.
nggDrawGetFontId	nggraph.h	drawctx, fontname, fontid	NG_EOK	Gets the font ID of a loaded font from its name.

API name	Source	Parameter	Return	Remark
nggDrawGetLineStyle	nggraph.h	drawctx	The line style used.	Gets the currently used line style.
nggDrawGetLineWidth	nggraph.h	drawctx	The line width used.	Gets the line width currently used for drawing boxes.
nggDrawGetOffset	nggraph.h	drawctx, offset	None.	Gets the offset currently used.
nggDrawGetPatOffset	nggraph.h	drawctx, patoffset	None.	Gets the pattern offset currently used.
nggDrawGetTextColor	nggraph.h	drawctx	The color used.	Gets the text color used.
nggDrawGetTextPat	nggraph.h	drawctx, textpat	None.	Gets the text pattern used.
nggDrawGetTransColor	nggraph.h	drawctx	The transparent color used.	Gets the transparent color used.
nggDrawGetUserClip	nggraph.h	drawctx, userclip	None.	Gets the user clipping currently used.
nggDrawImage	nggraph.h	drawctx, image, region, p	NG_EOK	Displays an image.
nggDrawLine	nggraph.h	drawctx, p1, p2	NG_EOK	Draws a line.
nggDrawMemGetDraw-Ctx	ngvid.h	dmemctx	The draw context associated.	Gets the draw context of a memory draw context.
nggDrawMemSurface	nggraph.h	dmemctx, region, orig	NG_EOK	Displays an off-screen surface used by the double buffering mechanism.
nggDrawPixel	nggraph.h	drawctx, p	NG_EOK	Draws a pixel.
nggDrawPolyLine	nggraph.h	drawctx, pts, nb_pts, ss	NG_EOK	Draws a polyline or a polygon.
nggDrawSetClip	nggraph.h	drawctx, clip	NG_EOK	Sets a clipping region.
nggDrawSetColor	nggraph.h	drawctx, c	NG_EOK	Sets the current colour value.
nggDrawSetFillColor	nggraph.h	drawctx, c	NG_EOK	Sets the filling colour value applied to the shapes.
nggDrawSetFillPatColor	nggraph.h	drawctx, c	NG_EOK	Sets the pattern colour value used to fill shapes.
nggDrawSetFillPatMode	nggraph.h	drawctx, drawmode	NG_EOK	Sets the filling pattern mode.

API name	Source	Parameter	Return	Remark
nggDrawSetFillPattern	nggraph.h	drawctx, fpattern	NG_EOK	Sets the pattern used to fill shapes.
nggDrawSetFontId	nggraph.h	drawctx, fontid	NG_EOK	Sets the current font ID.
nggDrawSetLineStyle	nggraph.h	drawctx, style	NG_EOK	Sets the line and border style used to draw shapes.
nggDrawSetLineWidth	nggraph.h	drawctx, width	NG_EOK	Sets the border thickness used when drawing boxes.
nggDrawSetMode	nggraph.h	drawctx, drawmode	NG_EOK	Sets the drawing mode.
nggDrawSetOffset	nggraph.h	drawctx, offset	NG_EOK	Sets the drawing offset.
nggDrawSetPalette	nggraph.h	drawctx, palette, start_index, nb_color	NG_EOK	Sets part of a palette in the current palette.
nggDrawSetPatOffset	nggraph.h	drawctx, patoffset	NG_EOK	Sets the pattern offset.
nggDrawSetTextColor	nggraph.h	drawctx, c	NG_EOK	Sets the filling colour value applied to text and bitmaps.
nggDrawSetTextPatColor	nggraph.h	drawctx, c	NG_EOK	Sets the pattern colour value used to fill text and bitmaps.
nggDrawSetTextPatMode	nggraph.h	drawctx, drawmode	NG_EOK	Sets the filling pattern mode applied to text.
nggDrawSetTextPattern	nggraph.h	drawctx, tpattern	NG_EOK	Sets the pattern used to fill text.
nggDrawSetTransparentColor	nggraph.h	drawctx, c	NG_EOK	Sets a transparent colour value.
nggDrawText	nggraph.h	drawctx, p, text, textlen, flags	NG_EOK	Draws a string in the current font.
nggEvMgrClose	nggevmgr.h	evmgrctx	NG_EOK	Closes a connection with the event manager.

API name	Source	Parameter	Return	Remark
nggEvMgrGetCursorPos	nggevmgr.h	evmgrctx, pos	NG_EOK	Gets the cursor position when the event manager module is used.
nggEvMgrGetDrawctx	nggevmgr.h	evmgrctx	The draw context used.	Gets the draw context from the event manager context.
nggEvMgrGetEvNb	nggevmgr.h	evmgrctx, evnb	NG_EOK	Gets the number of events queued.
nggEvMgrGetInfo	ngvid.h	evmgrctx, vidinfo	NG_EOK	Gets the frame buffer format when the event manager module is used.
nggEvMgrHideCursor	nggevmgr.h	evmgrctx	NG_EOK	Hides the cursor shape when the event manager module is used.
nggEvMgrLoadCursorShape	nggevmgr.h	evmgrctx, cursor	NG_EOK	Loads a cursor shape when the event manager module is used.
nggEvMgrMoveCursor	nggevmgr.h	evmgrctx, pos	NG_EOK	Moves the cursor shape when the event manager module is used.
nggEvMgrOpen	nggevmgr.h	evmgrctx, drawbuf, buflen	NG_EOK	Opens and initialises a connection with the event manager.
nggEvMgrRead	nggevmgr.h	evmgrctx, ev, evlen, mode	NG_EOK	Reads a user input in the event manager queue.
nggEvMgrSetClipList	nggevmgr.h	evmgrctx, cliplist, cliplen	NG_EOK	Sets the clipping list used by the video driver attached to the event manager.
nggEvMgrSetCursorShape	nggevmgr.h	evmgrctx, shape	NG_EOK	Sets a cursor shape when the event manager module is used.
nggEvMgrSetEvMask	nggevmgr.h	evmgrctx, mask	NG_EOK	Sets the event filter for a connection with the event manager.
nggEvMgrSetOrigin	nggevmgr.h	evmgrctx, orig	NG_EOK	Sets the origin of the drawing commands used by the video driver attached to the event manager.

API name	Source	Parameter	Return	Remark
nggVidDrawDraw	ngvid.h	dctx, buf, buf_len	NG_EOK	Executes the graphic requests stored in a buffer.
nggVidDrawGetDrawCtx	ngvid.h	dctx	The video draw context used.	Gets the draw context associated to a video draw context.
nggVidDrawGetInfo	ngvid.h	dctx, vidinfo	NG_EOK	Gets the frame buffer format.
nggVidDrawHideCursor	ngvid.h	dctx	NG_EOK	Hides the cursor.
nggVidDrawHideXorBox	ngvid.h	dctx	NG_EOK	Hides the cursor.
nggVidDrawInit	ngvid.h	dctx, vidp, drawbuf, buflen	NG_EOK	Initialises a draw context.
nggVidDrawLoadCusorShape	ngvid.h	dctx, cursor, shape	NG_EOK	Loads a new cursor in the cursor table.
nggVidDrawMemCtxInit	ngvid.h	dctx, dmemctx, surfbuf, surflen, off_screen, drawmbuf, buflen	NG_EOK NG_EINVAL NG_ENOSPC	Initialises a memory context to use the double buffering mechanism.
nggVidDrawMoveCursor	ngvid.h	dctx, pos	NG_EOK	Sets the cursor position
nggVidDrawSetBitBlit	ngvid.h	dctx, clip, p, box	NG_EOK	Copies a region from one point to another.
nggVidDrawSetClipList	ngvid.h	dctx, cliplist, cliplen	NG_EOK	Sets the clipping list.
nggVidDrawSetCursor	ngvid.h	dctx, cursor	NG_EOK	Loads and sets a cursor shape.
nggVidDrawSetCursorShape	ngvid.h	dctx, shape	NG_EOK	Sets a cursor shape.
nggVidDrawSetOrigin	ngvid.h	dctx, orig	NG_EOK	Sets the origin of drawing commands.
nggVidDrawShowCursor	ngvid.h	dctx	NG_EOK	Shows the cursor.
nggVidDrawShowXorBox	ngvid.h	dctx, xorbox, clipping	NG_EOK	Displays a box using the XOR method.

8.4 Sample Program for NexGenGRAPH

Test Program: LCDDIAG

Makefile directionary: $(ADVEVA)/i186/apps/samples/lcddiag

Purpose: Demonstrate LCD functions (e.g. adjust LCD, draw a pixel, line, circle, box...etc.)

Graph test program includes the shell commands presented below. You can use HELP to display the list of shell command.

Standard Commands

Command	Description
help	Display list of commands and help information
ver	Display version of NexGenOS.
netstat	Display information about the network drivers and protocols
devstat	Display information about the configuration of installed serial devices and their statistics.
ifconfig	Display information and configures network interfaces
sysctl	Display and configure system configuration.
route	Display routing table.
lsmod	Display list of loaded module in ROM Monitor sample program.
exit	Terminate the shell session.
ping	Sends ICMP echo requests to a host. This command can be used to verify the IP communication between two hosts.
arp	Display Address Resolution Protocol host table.
diag	Self-diagnostic command which includes below two functions: • Adjust touch screen of the LCD panel. • Diagnose general purpose GPIO, I/O, UART port in the EVA-X1610 evaluation board.

Graph command

Command	Description
cls	Clean the screen.
dflush	Execute the drawing requests.
dsetcolor	Set drawing color.
dsetfillcolor	Set filling color.
dsettextcolor	Set the text and bitmap color.
dpixel	Draw a pixel.
dline	Draw a line.
dbox	Draw a box.
dcircle	Draw a circle/ellipse.
darc	Draw an arc of circle/ellipse.
dpolyline	Draw a polyline/polygon.
getptchar	Get the pointed character in a string.

[Demonstration]

When you turn on the EVA-X1610C evaluation board with LCD panel, please follow below setup procedure.

System setup

<Step1>Set drawing color.

```
>setcolor 255 255 255 0
Set the drawing color to RGB = 255-255-255
```

<Step2> Set filling color

```
>dsetfillcolor 255 255 255 0
Set the filling color to RGB = 255-255-255
```

<Step3> Set the text color.

```
>dsettextcolor 255 255 255 0
Set the drawing text color to RGB = 255-255-255
```

[Self-Diagnostic]

To adjust the touch screen of LCD panel, you can use following commands.

- diag adjts Adjust touch screen of LCD panel without accuracy check.
- diag testts Adjust touch screen of LCD panel with accuracy check.

To ensure the evaluation board is okay, you can use following commands to test your evaluation board.

- diag testgpio Test all of the General Purpose I/O ports.
- diag testuart Test COM2 (UART1) on the evaluation board.
- diag testall Test GPIO, I/O, UART1 sequentially on the evaluation board.

NOTE:

If you want to test a specified COM port, you must plug the loop back communication port on that port.

For example, following is the results for execute the command "diag testall", the system will adjust touch screen of LCD and test GPIO, I/O UART0 and UART1 sequentially.

```
>diag testall
————————————[Backlight]————————————
Backlight turn on
————————————————[GPIO]————————————————
GPIO test tool.
Turn off all GPIO LEDs ...
Turn on GPIO LED one by one ...
[0][1][7][8][9][10][11][12][13][15][25][26][27][28][31]
Turn off GPIO LED one by one ...
[0][1][7][8][9][10][11][12][13][15][25][26][27][28][31]
Flash all GPIO LEDs 2 times
Finished.
————————————————[UART]————————————————
UART1 self diagnostic ...
[DTR, pass] [RTS, pass] [CTS, fail] [DSR, fail] [CD, fail]7
115200, 8-bit, 2-stop,  odd : pass
```

```
————————[Touch Screen]————————
[Try 1] ...
Adjust touch screen...
Please click on the 1st cursor hot point....
Please click on the 2nd cursor hot point....
Please click on the 3rd cursor hot point for examing....
Finished.
————————[Backlight]————————
Backlight turn off
>
```

[Drawing]

To draw a graph on the LCD, you can use following commands:

- dpixel Draw a pixel.
- dbox Draw a box.
- dcircle Draw a circle/ellipse.
- darc Draw an arc of circle/ellipse.
- dpolyline Draw a polyline/polygon.

Example1: Draw a line.

```
>dline 10 10 100 100
Drawing Line.
  >dflush
  Flush draw buffer.
```

The result will be shown on the LCD panel as below.

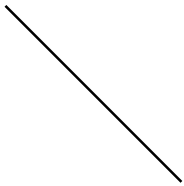


Fig 8-4: Draw a line on the LCD panel

Example2: Draw a box.

```
>dbox 100 100 100 3
Drawing Box.
>dflush
Flush draw buffer.
```

The result will be shown on the LCD panel as below.

Fig 8-5: Draw a box on the LCD panel

NOTE

To know the detail of the specific command, you can follow below syntax:

HELP *command name*

Example

If you want to know the detail of the command “dbox”, you can:

```
>help dbox
x y w h flags - Draw a box.
  flag=1 : border
  flag=2 : filling
  flag=3 : border + filling
```