Eddy DK

Programmer Guide

Ver 2.5.3.1

2013.01.21

Revision History

Revision Date	Document Version	Pages	Description
Feb-5-2009	2.1.0.1	All	Initial released
Sep-10-2009	2.1.0.2	All	Added Eddy-WiFi
Nov-12-2009	2.1.0.3	12	J2 pin33 PC12 → PC13 J2 pin35 PC13 → PC12
Oct-22-2009	2.1.0.3	17,18,19	J2 pin33 PC12 → PC13 J2 pin35 PC13 → PC12
		18,19	J2 pin33 J9_26 → J9_33 J2 pin34 J9_25 → J9_34 J2 pin33 J9_24 → J9_35
Nov-23-2009	2.1.0.3	2,4,6	Added S4M
Jun-25-2010	2.1.1.1	All	Open Linux Version Added Eddy-BT
Sep-15-2010	2.5.1.1	2,9	Added Eddy-CPU v2.5
Jan-20-2011	2.5.1.1		Added Eddy-S4M v2.5
Feb-15-2011	2.5.1.1		Added Eddy-CPU/mp v2.5
Aug-09-2011	2.5.1.1		Added Eddy-CPU/mp 32bit v2.5
Dec-09-2011	2.5.1.1		Added Eddy-WIFI v3.0
Jan-21-2013	2.5.3.1		New Firmware (Linux kernel v2.6.30)

Table of Contents

Chapter 1. Introduction

This chapter covers about introduction and development process regarding Eddy-DK v2.1 and Eddy-S4M-DK v2.1.

1.1 About this manual

This manual covers about how to develop a user application and how to apply it in Eddy DK (Eddy-DK, Eddy-S4M-DK). Additionally, information regarding understanding the OS in Eddy module and API functions are supplied.

After reading this guide, a programmer should be able to write his or her own application and execute it using Eddy DK.

1.2 Notice to readers

This document is designed for programmers who wish to develop a new application using Eddy DK. It is strongly recommended that the programmer read this document before starting any programming. If you are an administrator or an end user who just needs to apply the module into practical applications, you do not need to read this document. User's Guide will be helpful in that case. This guide covers the complete process of building a user file system from writing source codes to making a firmware that can be uploaded and executed on Eddy module.

1.3 Manual contents

Chapter 1. Introduction is a preface with general information and notices..

Chapter 2. Getting Started gives brief information needed before starting programming.

Chapter 3. Development Environment explains about the process of writing a customized application and related work.

Chapter 4. Compile an Application Program covers the process of compiling an application with a makefile.

Chapter 5. Create User File System tells you how to convert a compiled application to a compressed ram disk form and apply it to Eddy module.

Chapter 6. Bootloader covers commands used for bootloader and how to display and show the environment variables.

Chapter 7. Library explains about libraries and API functions that you can use while programming an application.

Chapter 8. Eddy Software shows how to implement simple TCP/IP and serial communication routines using example source codes included in the DK (Development Kit).

Chapter 9. Handling HTML & CGI covers a guide for integrating your own applications with web interface provided by Eddy DK**.**

Chapter 10. Appendix provides information regarding building your own applications through Eddy DK, how to recover the system and utilities for upgrading the firmware.

1.4 Eddy DK related documents

The following tables show Eddy DK related documents.

Document Name	Description
User guide	Manual for Eddy user; how to set connection with Eddy, status monitoring, firmware updating, and other tasks regarding management are introduced in this guide.
Programmer's guide	Description regarding a compilation required by a programmer to run an application in Eddy, linking, creating user file system and how to upload it to Eddy, API functions for customized application are included in this guide.
LemonIDE manual	It covers how to write and compile source codes for Windows, remote debugging, and creating user file system under Eclipse based IDE for developing Eddy embedded software.

If you need a brief information about Eddy or embedded device servers in general, please visit our corporate website at http://www.sysbas.com/ or Eddy Developer's community at http://www.embeddedmodule.com/. You can view and download documents related to Eddy as well as latest software and firmware updates.

Other Eddy related resources are as follows:

Document name	**Description**
Eddy-CPU Spec Sheet	Specifications about Eddy-CPU and DK board
Eddy-S4M Spec Sheet	Specification about Eddy-S4M
Eddy-WiFi Spec Sheet	Specification about Eddy-Wi-Fi
Eddy-BT Spec Sheet	Specification about Eddy-BT
LemonIDE Spec Sheet	Integrated Development Environment
Eddy White Paper	An introductory reading for anyone new to embedded device server. Covers background, history, market environment and technology.

All documents are updated promptly, so check for the recent document update. The contents in these documents are subject to change without any notice in advance.

1.5 Technical Support

There are three ways you can get a technical support from SystemBase.

First, visit our website at http://www.solvline.com/ there you can read FAQs or post an inquiry.

Second, you can email us at tech@sysbas.com. Any kind of inquiries, requests, and comments are welcome.

Finally, you can call us at the customer center for immediate support. Our technical support team will kindly help you get over with the problem.

Homepage: http://www.sysbas.com/
Tel: +82-2-855-0501
Fax: +82-2-855-0580
16F, Daerung Post Tower 1, 212-8, Guro-dong, Guro-gu, Seoul, Republic of Korea
Postal code: 152-790

Chapter 2. Getting Started

This chapter covers about package and installation, and discusses key features in Eddy-DK.

2.1 What can you do with Eddy DK?

Eddy DK is designed to help programmers to develop a customized application that can be applied to Eddy module easier and faster. It has been a time-consuming and burdensome work to port an operating system and develop an application on a new hardware. Eddy module and Software Development Kit makes this work easier.

Eddy DK is different with other device servers in which it can run customized applications. Users can upload most existing socket/serial communication applications that are running on the Linux environment. This openness allows users to apply wide variety of functions into the module with relatively less restrictions.

Eddy DK supports IDE (LemonIDE) and SDK environment to help programmers to execute their own applications on the module. Programmers can easily write applications using the Linux environment, with the help of SDK and example source codes. Cross-compiler running on Linux environment helps your applications to run on the Eddy module smoothly. Embedded Linux on Eddy can provide stable and rapid environment for your applications.

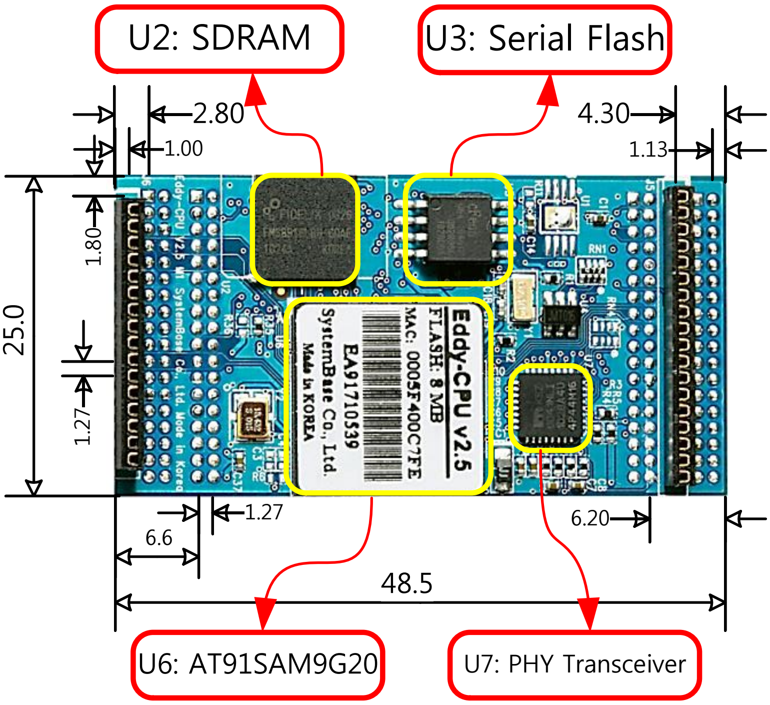
2.2 Contents in Eddy DK

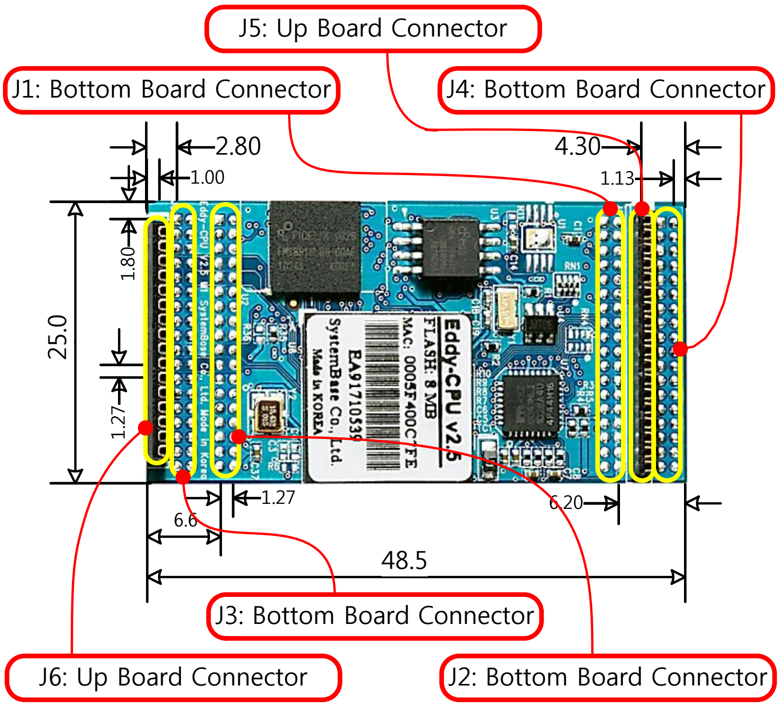
When you purchase an Eddy DK product, one unit of Eddy module is included with the DK board.
Eddy DK package contains as follows. Please make sure all contents are included.

For Eddy-DK,	(1 unit of Eddy-CPU v2.1/v2.5, 1 unit of Eddy-DK v2.1 board)
For Eddy Eddy-S4M-DK	(1 unit of Eddy-S4M v2.5, 1 unit of Eddy-S4M-DK board, (Option: Eddy-S4M-JIG))

1 Serial cable
1 LAN cable
1 USB type A to type B cable
1 Power adapter
1 CD (SystemBase SDK, LemonIDE, Compile Environment, Utility, Manuals)

2.3 Eddy-CPU v2.1 / v2.5





* Eddy-CPU v2.1 / v2.5 Pin Assignment

J1			
Pin	Signal Name	Pin	Signal Name
1	PA5	2	PA4
3	PC5	4	PC19
5	PC21	5	PC23
7	HDMA	8	NC
9	HDPA	10	DDM
11	PC26	12	DDP
13	PC4 (RDY#)	14	PC16
15	ICE_NTRST	16	RTCK
17	TDO	18	TMS
19	TDI	20	TCK
21	3.3V	22	GND
23	3.3V	24	GND
25	PB29 (CTS1)	26	PB28 (RTS1)
27	PB6 (TXD1)	28	PB7 (RXD1)
29	A20	30	A19
31	LAN_Speed	32	LAN_Link
33	LAN_RX-	34	LAN_RX+
35	LAN_TX-	36	LAN_TX+

J2			
Pin	Signal Name	Pin	Signal Name
1	A15	2	A14
3	A13	4	A12
5	A11	5	A10
7	A9	8	A8
9	A7	10	A6
11	A5	12	A4
13	A3	14	A2
15	A1	16	A0
17	PC9	18	NWE
19	FPG	20	NRD
21	GND	22	3.3V
23	GND	24	3.3V
25	D7	26	D6
27	D5	28	D4
29	D3	30	D2
31	D1	32	D0
33	PC13	34	JTAGSEL
35	PC12	36	NC

J3			
Pin	Signal Name	Pin	Signal Name
1	PID0	2	PID1
3	PID2	4	PID3
5	PID4	5	GND
7	PC14	8	PC17
9	PC18	10	PC8 (RTS3)
11	PC20	12	PC10 (CTS3)
13	PA22	14	PC15 (IRQ1)
15	PB8	16	PB9 (RXD2)
17	PB10	18	PB11(RXD3)
19	PC0	20	PC1 (AD1)
21	PC2	22	PC3 (AD3)
23	PB14 (DRXD)	24	PB15 (DTXD)
25	GND	26	GND
27	BMS	28	NRST
29	PB23 / DCD0	30	PB5 / RXD0
31	PB4 / TXD0	32	PB24 / DTR0
33	PB22 / DSR0	34	PB26 / RTS0
35	PB27 / CTS0	36	PB25 / RI0

J4			
Pin	Signal Name	Pin	Signal Name
1	PB12	2	PB13
3	PB30	4	PB31
5	PB0	5	PC22
7	PB1	8	PB16
9	PB2	10	PB17
11	PB3	12	PB18
13	BHDM	14	PB19
15	BHDP	16	PB20
17	A16	18	PB21
19	A17	20	A18
21	D8	22	D9
23	D10	24	D11
25	D12	26	D13
27	D14	28	D15
29	TWD	30	TCK
31	NANDOE	32	NAND_CLE /
33	NANDWE	34	NAND_ALE /
35	NC	36	NC

J5	
Pin	Signal Name
1	PB0
2	PB1
3	PB2
4	PB3
5	3.3V
6	3.3V
7	BHDM, USB Host Data(-)
8	BHDP, USB Host Data(+)
9	PA31 / TXD4
10	PA30 / RXD4
11	NRST
12	GND
13	GND
14	PA9 / WPID0
15	PC6 / WPID1
16	PC7 / WPID2
17	NC
18	NC

J6	
Pin	Signal Name
1	NC
2	NC
3	3.3V
4	3.3V
5	PC25 / BT_Factory
6	PB10 / TXD3
7	PB11 / RXD3
8	PC8 / RTS3
9	PC10 / CTS3
10	PC24 / BT_MODE
11	NRST
12	GND
13	GND
14	NC
15	NC
16	NC

J1 Specifications

J1			
Pin	Signal Name	Pin	Signal Name
1	PA5	2	PA4
3	PC5	4	PC19
5	PC21	5	PC23
7	HDMA	8	NC
9	HDPA	10	DDM
11	PC26	12	DDP
13	PC4 (RDY#)	14	PC16 (nRESET)
15	ICE_NTRST	16	RTCK
17	TDO	18	TMS
19	TDI	20	TCK
21	3.3V	22	GND
23	3.3V	24	GND
25	PB29 (CTS1)	26	PB28 (RTS1)
27	PB6 (TXD1)	28	PB7 (RXD1)
29	A20	30	A19
31	LAN_Speed	32	LAN_Link
33	LAN_RX-	34	LAN_RX+
35	LAN_TX-	36	LAN_TX+

J1 Pin Description

Pin No.	Name	DK v2.1 Pin No	Expansion Header Pin No.	Description	
1	PA5	J10_1	J4_2	Peripheral A : CTS2	UART #2 Clear to Send Signal
				Peripheral B : MCBD1	Cannot be used. In Eddy-CPU v2.1/ v2.5, data flash is used which is connected to SPI0. Therefore, MCDB0, MCDB3, MCCDB signals used for SPI0 and multiplexing cannot be used thus making multimedia card slot B unable to be used.
2	PA4	J10_2	J4_1	Peripheral A : RTS2	UART #2 Request to Send Signal
				Peripheral B : MCDB2	Cannot be used.
3	PC5	J10_3	J4_12	Peripheral A : A24	External Address Bus
				Peripheral B : SPI1_NPCS1	SPI1(Serial Peripheral Interface) Peripheral Chip Select 1
4	PC19	J10_4	J4_24	Peripheral A : A24	Multimedia Card Slot B Data
				Peripheral B : SPI1_NPCS2	SPI1(Serial Peripheral Interface) Peripheral Chip Select 2
5	PC21	J10_5	J4_26	Peripheral A : D21	External Data bus
				Peripheral B : EF100	Ethernet(WAN) Force 100Mbit/sec.
6	PC23	J10_6	J4_28	Peripheral A : D23	External Data Bus
7	HDMA	J10_7	J1_27	USB Host Port A Data -	
8	NC	J10_8	--	Not Connect	
9	HDPA	J10_9	J1_29	USB Host Port A Data +	
10	DDM	J10_10	-	USB Device Port Data -	

11	PC26	J10_11	-	D26	External Data Bus
12	DDP	J10_12	-	USB Device Port Data +	
13	PC4 (RDY#)	J10_13	J4_11	Eddy DK v2,1 : RDY#(OUT)	Ready signal. Output signal for CPU operation status
				Peripheral A : A23	External Address Bus
				Peripheral B : SPI1_NPCS2	SPI1(Serial Peripheral Interface) Peripheral Chip Select 2
14	PC16 (nRESET)	J10_14	J4_21	Eddy DK v2,1 : nRESET#(IN)	Continually perform polling input signal from external reset key, check the duration of "Low" implement by S/W as following. Less than 5seconds : General reset Same or more than 5seconds : Factory default
				Peripheral A : D16	External Data Bus
				Peripheral B : SPI0_NPCS2	Cannot be used. In SPI0, SPI0_SPCK, SPI0_MISO, SPI0_MOSI signals are not connected to external therefore cannot be used.
15	ICE_NTRST	J10_15	J7_3	ICE Test Reset Signal	
16	RTCK	J10_16	J7_11	Return Test Clock	
17	TDO	J10_17	J7_13	Test Data Out	
18	TMS	J10_18	J7_7	Test Mode Select	
19	TDI	J10_19	J7_5	Test Data In	
20	TCK	J10_20	J7_9	Test Clock	
21	3.3V	3.0V to 3.6V power input			

<table>
<tr><td>22</td><td>GND</td><td colspan="4">Ground</td></tr>
<tr><td>23</td><td>3.3V</td><td colspan="4">3.0V to 3.6V power input</td></tr>
<tr><td>24</td><td>GND</td><td colspan="4">Ground</td></tr>
<tr><td rowspan="2">25</td><td rowspan="2">PB29</td><td rowspan="2">J10_25</td><td rowspan="2">J2_30</td><td>Peripheral A : CTS1</td><td>USART1 Clear To Send</td></tr>
<tr><td>Peripheral B : ISI_VSYNC</td><td>Image Sensor Vertical Synchronization</td></tr>
<tr><td rowspan="2">26</td><td rowspan="2">PB28</td><td rowspan="2">J10_26</td><td rowspan="2">J2_29</td><td>Peripheral A : RTS1</td><td>USART1 Request To Send</td></tr>
<tr><td>Peripheral B : ISI_PCK (IN)</td><td>Image Sensor Pixel Clock Provided by the Image Sensor</td></tr>
<tr><td rowspan="2">27</td><td rowspan="2">PB6</td><td rowspan="2">J10_27</td><td rowspan="2">J2_7</td><td>Peripheral A : TXD1</td><td>USART1 Transmit Data</td></tr>
<tr><td>Peripheral B : TCLK1</td><td>Timer Counter ch1 External CLK IN</td></tr>
<tr><td rowspan="2">28</td><td rowspan="2">PB7</td><td rowspan="2">J10_28</td><td rowspan="2">J2_8</td><td>Peripheral A : RXD11</td><td>USART1 Receive Data</td></tr>
<tr><td>Peripheral B : TCLK2</td><td>Timer Counter ch2 External CLK IN</td></tr>
<tr><td colspan="6"><b>Address Bus</b></td></tr>
<tr><td>29</td><td>A20</td><td>J10-29</td><td>J1_31</td><td colspan="2">Address Bus</td></tr>
<tr><td>30</td><td>A19</td><td>J10_30</td><td>J1_32</td><td colspan="2">Address Bus</td></tr>
<tr><td colspan="6"><b>Ethernet 10/100 (Auto MDI/MDIX)</b></td></tr>
<tr><td>31</td><td>LED_Speed</td><td>J10_31</td><td>-</td><td colspan="2">LAN connection speed
<table>
<tr><th>Speed</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>10Base-T</td><td>H</td><td>OFF</td></tr>
<tr><td>100Base-TX</td><td>L</td><td>ON</td></tr>
</table></td></tr>
<tr><td>32</td><td>LED_Link</td><td>J10_32</td><td>-</td><td colspan="2">LAN connection status
<table>
<tr><th>Link/Activity</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>No Link</td><td>H</td><td>OFF</td></tr>
<tr><td>Link</td><td>L</td><td>ON</td></tr>
<tr><td>Activity</td><td>Toggle</td><td>Blinking</td></tr>
</table></td></tr>
<tr><td>33</td><td>LAN_RX-</td><td>J10_33</td><td>-</td><td colspan="2">Physical receive or transmit signal (- differential) of internal CPU Ethernet PHY (WAN)</td></tr>
</table>

34	LAN_RX+	J10_34	-	Physical receive or transmit signal (+ differential) of internal CPU Ethernet PHY (WAN)
35	LAN_TX-	J10_35	-	Physical transmit or receive signal (- differential) of internal CPU Ethernet PHY (WAN)
36	LAN_TX+	J10_36	-	Physical transmit or receive signal (+ differential) of internal CPU Ethernet PHY (WAN)

J2 Specifications

J2			
Pin	Signal Name	Pin	Signal Name
1	A15	2	A14
3	A13	4	A12
5	A11	5	A10
7	A9	8	A8
9	A7	10	A6
11	A5	12	A4
13	A3	14	A2
15	A1	16	A0
17	PC9	18	NWE
19	FPG	20	NRD
21	GND	22	3.3V
23	GND	24	3.3V
25	D7	26	D6
27	D5	28	D4
29	D3	30	D2
31	D1	32	D0
33	PC12	34	JTAGSEL
35	PC13	36	NC

J2 Pin Description

Pin No.	Name	DK v2.1 Pin No.	Expansion Header Pin No.	Description	
1~16	A[15:0]	J9_1 -J9_16	J3_4-J3_20	External Address Bus 0-15 (0 at reset) CPU and DK are directly connected but with J3, it is connected through a buffer.	
17	PC9	J9_17	J4_14	Peripheral A : NCS5	External device Chip Select 5. 256MB memory area addressable, active low
				Peripheral B : TIOB0	Timer Counter ch0 I/O Line B
18	NEW	J9_18	J1_21	External device Write Enable signal, active low	
19	FPG	J9_19	-	For Flash Programming Data flash in Eddy-CPU v2.1/V2.5 USB port is used to program boot code (loader, kernel, file system). For detailed information, refer to 2.4.2.3 S6: NAND Flash & Data Flash Chip Select.	
20	NRD	J9_20	J1_23	External device Read Enable signal, active low	
21, 23	GND	Ground			
22, 24	3.3V	3.0V to 3.6V power input			
25~32	D[7:0]	J9_25 - J3_32	J3_29 - J3_36	External Data Bus 0-7 CPU and DK are directly connected but with J3, it is connected through a buffer. In order to activate the buffer, PC13 (NCS6: Chip Select 6) needs to be enabled. While reset, it is operated as pulled-up input.	
33	PC12	J9_24	J4_17	Peripheral A : IRQ0	External Interrupt Input 0
				Peripheral B : NCS7	External device Chip Select 7. 256MB memory area addressable, active low
34	JTAGSEL	J9_25	-	JTAG boundary scan can be used by connecting pin34 and 36(connect J14). This pin should not be connected when using ICE (In-Circuit Emulator) or in normal operation status.	

35	PC13	J9_26	J4_18	Eddy-DK v2.1 : NCS6	Data bus connected to an extended header can be used when NCS6 is enabled.
				Peripheral A : FIQ	Fast Interrupt Input
				Peripheral B : NCS6	External device Chip Select 6 256MB memory area addressable, active low
36	NC	Not Connect			

J3 Specifications

J3			
Pin	Signal Name	Pin	Signal Name
1	PID0	2	PID1
3	PID2	4	PID3
5	PID4	5	GND
7	PC14	8	PC17
9	PC18	10	PC8 (RTS3)
11	PC20	12	PC10 (CTS3)
13	PA22	14	PC15 (IRQ1)
15	PB8	16	PB9 (RXD2)
17	PB10	18	PB11(RXD3)
19	PC0	20	PC1 (AD1)
21	PC2	22	PC3 (AD3)
23	PB14 (DRXD)	24	PB15 (DTXD)
25	GND	26	GND
27	BMS	28	NRST
29	PB23 / DCD0	30	PB5 / RXD0
31	PB4 / TXD0	32	PB24 / DTR0
33	PB22 / DSR0	34	PB26 / RTS0
35	PB27 / CTS0	36	PB25 / RI0

J3 Pin Description

Pin No.	Name	DK v2.1 Pin No	Expansion Header Pin No.	Description	
1-5	PID[4:0]	J8_1 ~J8_5	-	Product ID only used by the manufacturer. Please do not work on these pins.	
6,25,26	GND	Ground			
7	PC14	J8_7	J4_19	Peripheral A : NCS3	External Device Chip Select 3
				Peripheral B : IRQ2	External Interrupt Input 2
8	PC17	J8_8	J4_22	Peripheral A : D17	External Data Bus
				Peripheral B : SPI0_NPCS3	Cannot be used
9	PC18	J8_9	J4_23	Peripheral A : D18	External Data Bus
				Peripheral B : SPI1_NPCS1	SPI1(Serial Peripheral Interface) Peripheral Chip Select 1
10	PC8	J8_10	J4_13	Peripheral A : NCS4	External Device Chip Select 4
				Peripheral B : RTS3	USART3 Request to Send
11	PC20	J8_11	J4_25	Peripheral A : D20	External Data Bus
				Peripheral B : SPI1_NPCS3	SPI1(Serial Peripheral Interface) Peripheral Chip Select 3
12	PC10	J8_12	J4_15	Peripheral A : A25	External Address Bus
				Peripheral B : CTS3	USART3 Clear to Send

13	PA22	J8_13	-	Digital I/O Input 4	
14	PC15	J8_14	J4_20	Peripheral A : NWAIT	External Wait Signal Input
				Peripheral B : IRQ1	External Interrupt Input 2
15	PB8	J8_15	J2_9	Peripheral A : TXD2	UART2 Transmit Data
16	PB9	J8_16	J2_10	Peripheral A : RXD2	UART2 Receive Data
17	PB10	J8_17	J2_11	Peripheral A : TXD3	UART3 Transmit Data
				Peripheral B : ISI_D8	Image Sensor Data 8
18	PB11	J8_18	J2_12	Peripheral A : RXD3	UART3 Receive Data
				Peripheral B : ISI_D9	Image Sensor Data 9
19	PC0	J8_19	J4_7	Peripheral A : AD0	Analog to Digital Converter Input Ch0
				Peripheral B : SCK3	USART3 Serial Clock
20	PC1	J8_20	J4_8	Peripheral A : AD1	Analog to Digital Converter Input Ch1
				Peripheral B : PCK0	Programmable Clock Output 0
21	PC2	J8_21	J4_9	Peripheral A : AD2	Analog to Digital Converter Input Ch2
				Peripheral B : PCK1	Programmable Clock Output 1
22	PC3	J8_22	J4_10	Peripheral A : AD3	Analog to Digital Converter Input Ch3
				Peripheral B : SPI1_NPCS3	SPI1(Serial Peripheral Interface) Peripheral Chip Select 3
23	PB14	J8_23	J2_15	Peripheral A : DRXD	Debug Receive Data
24	PB15	J8_24	J2_16	Peripheral A : DTXD	Debug Transmit Data

27	BMS	J8_27	-	Boot Mode Select signal BMS = 1, Boot on Embedded ROM BMS = 0, Boot on External Memory	
28	NRST	J8_28	J1_20	External device Reset signal, active low signal	
29	PB23	J8_29	J4_28	Peripheral A : DCD0	USART0 Data Carrier Detection
				Peripheral B : ISI_D3	Image Sensor Data 3
30	PB5	J8_30	J2_6	Peripheral A : RXD0	USART0 Receive Data
31	PB4	J8_31	J2_5	Peripheral A : TXD0	USART0 Transmit Data
32	PB24	J8_32	J2_25	Peripheral A : DTR0	USART0 Data Terminal Ready
				Peripheral B : ISI_D4	Image Sensor Data 4
33	PB22	J8_33	J2_23	Peripheral A : DSR0	USART0 Data Set Ready
				Peripheral B : ISI_D2	Image Sensor Data 2
34	PB26	J8_34	J2_27	Peripheral A : RTS0	USART0 Request To Send
				Peripheral B : ISI_D6	Image Sensor Data 6
35	PB27	J8_35	J2_28	Peripheral A : CTS0	USART0 Clear To Send
				Peripheral B : ISI_D7	Image Sensor Data 7
36	PB25	J8_36	J2_26	Peripheral A : RI0	USART0 Ring Indicator
				Peripheral B : ISI_D5	Image Sensor Data 5

J4 Specifications

J4			
Pin	Signal Name	Pin	Signal Name
1	PB12	2	PB13
3	PB30	4	PB31
5	PB0	5	PC22
7	PB1	8	PB16
9	PB2	10	PB17
11	PB3	12	PB18
13	BHDM	14	PB19
15	BHDP	16	PB20
17	A16	18	PB21
19	A17	20	A18
21	D8	22	D9
23	D10	24	D11
25	D12	26	D13
27	D14	28	D15
29	TWD	30	TCK
31	NANDOE	32	NAND_CLE /
33	NANDWE	34	NAND_ALE /
35	NC	36	NC

J4 Pin Description

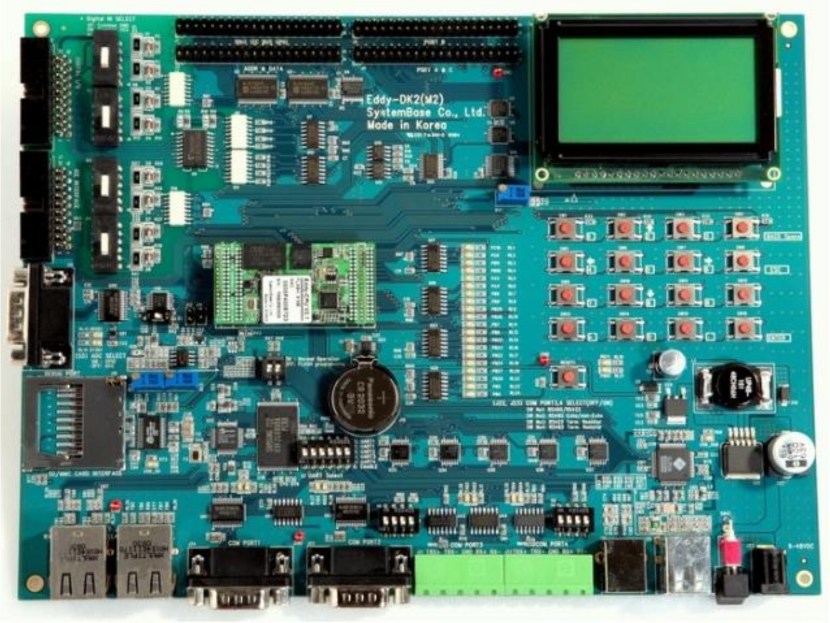
Pin No.	Name	DK v2.1 Pin No	Expansion Header Pin No.	Description	
1	PB12	J11_1	J2_17	Peripheral A : TXD5	USART5 Transmit Data
				Peripheral B : ISI_D10	Image Sensor Data 10
2	PB13	J11_2	J2_18	Peripheral A : RXD5	USART5 Receive Data
				Peripheral B : ISI_D11	Image Sensor Data 11
3	PB30	J11_3	J2_31	Peripheral A : PCK0	Programmable Clock Output 0
				Peripheral B : ISI_HSYNC	Image Sensor Horizontal Synchronization
4	PB31	J11_4	J2_32	Peripheral A : PCK1	Programmable Clock Output 1
5	PB0	J11_5	J2_2	Peripheral A : SPI1_MISO	SPI1(Serial Peripheral Interface) Master In Slave Out
				Peripheral B : TIOA3	Timer Counter ch3 I/O Line A
6	PC22	J11_6	J4_27	Peripheral A : D22	
				Peripheral B : TCLK5	Timer Counter ch5 External CLK IN
7	PB1	J11_7	J2_3	Peripheral A : SPI1_MOSI	
				Peripheral B : TIOB3	Timer Counter ch3 I/O Line B
8	PB16	J11_8	J2_17	Peripheral A : TK0	SSC Transmit Clock
				Peripheral B : TCLK3	Timer Counter ch3 External CLK IN
9	PB2	J11_9	J2_4	Peripheral A : SPI1_SPCK	SPI1(Serial Peripheral Interface) Serial Clock

				Peripheral B : ISI_D3	Image Sensor Data 3
10	PB17	J11_10	J2_18	Peripheral A : TF0	SSC Transmit Frame Sync
				Peripheral B : TCLK4	Timer Counter ch4 External CLK IN
11	PB3	J11_11	J2_5	Peripheral A : SPI1_NPCS0	SPI1 (Serial Peripheral Interface) Peripheral Chip Select 0
				Peripheral B : TIOA5	Timer Counter ch5 I/O Line A
12	PB18	J11_12	J2_19	Peripheral A : TD0	SSC Transmit Data
				Peripheral B : TIOB4	Timer Counter ch4 I/O Line B
13	HDMB	J11_13	J1_28	USB Device Port Data -	
14	PB19	J11_14	J2_20	Peripheral A : RD0	SSC Receive Data
				Peripheral B : TIOB5	Timer Counter ch5 I/O Line B
15	HDPB	J11_15	J1_30	USB Device Port Data +	
16	PB20	J11_16	J2_21	Peripheral A : RK0	SSC Receive Clock
				Peripheral B : ISI_D0	Image Sensor Data 0
17	A16	J11_17	J3_3	External Address Bus	
18	PB21	J11_18	J2_22	Peripheral A : RF0	SSC Receive Frame Sync
				Peripheral B : ISI_D1	Image Sensor Data 1
19	A17	J11_19	J3_2	External Address Bus	
20	A18	J11_20	J3_1		
21-28	D[8:15]	J11_21 ~J11_28	J3_21 ~J3_28	External Data Bus 8-15 CPU and DK are directly connected but with J3, it is connected through a buffer. In order to	

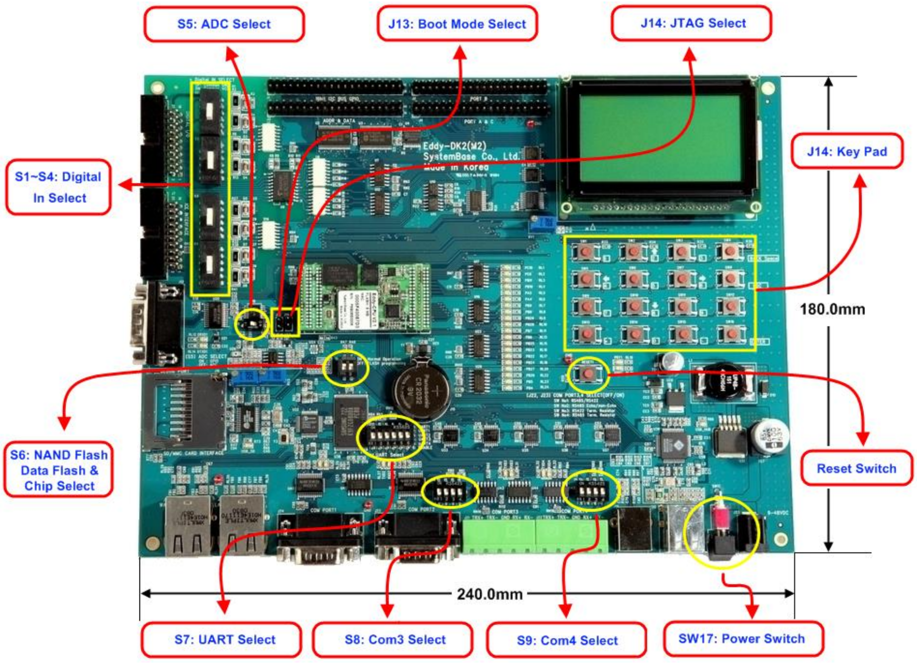
				activate the buffer, PC13 (NCS6: Chip Select 6) needs to be enabled. While reset, it is operated as pulled-up input.
29	TWD	J11_29	J4_3	Two-wire Serial Data. This pin cannot be used as GPIO.
30	TWCK	J11_30	J4_4	Two-wire Serial Data. This pin cannot be used as GPIO.
31	NANDOE	J11_31	-	NAND Flash Output Enable
32	A22	J11_32	J1_29	Address Bus CPU and DK are directly connected but with J3, it is connected through a buffer.
33	NANDWE	J11_33	-	NAND Flash Write Enable
34	A21	J11_34	J1_30	Address Bus
35,36	NC	Not Connect		

2.4 Eddy-DK v2.1

2.4.1 Product Image



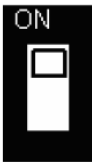
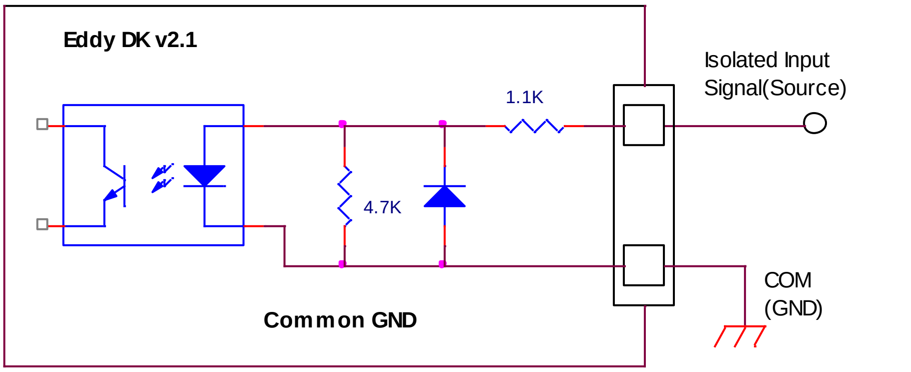
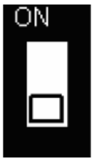
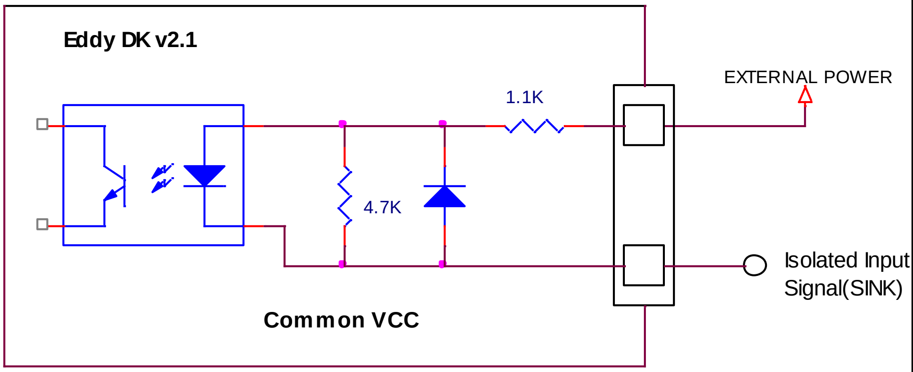
2.4.2 Switch Description



S1~S4: Digital In Select

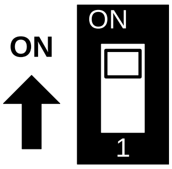
If S1~S4 switches are set, 2 mode can be selected for digital input. If you set switch as following connected device will operate as GND common mode or VCC common mode. Digital input circuit in DK is only for your reference, in real life use, you must designing it with considering the voltage and current.

Common input setting (Common for S1~S4)

MODE	Switch	Description
GND Common	UP 	 Eddy DK v2.1 1.1K 4.7K Isolated Input Signal(Source) COM (GND) Common GND
VCC Common	Down 	 Eddy DK v2.1 1.1K 4.7K EXTERNAL POWER Isolated Input Signal(SINK) Common VCC

2.4.2.1 S5: ADC Select

Determine how switch PC0-PC4 will be used as. By temperature sensor and illuminance sensor in DK, selection can be made whether to use an analog input or GPIO connected through expansion header.



SW OFF: use ADC
SW ON: use GPIO

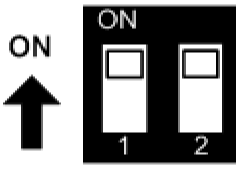
Pin	Feature	Used for	I/O
PC0	ADC0	Input temperature sensor(LM50), RN: U22	IN
PC1	ADC1	Input illuminance sensor (BH1600), RN: U26	IN

PC2	ADC2	Input temperature sensor (TMP300), RN: U24	IN
PC3	ADC3	N/A	IN

* RN = Reference Number

2.4.2.2 S6:NAND Flash & Data Flash Chip Select

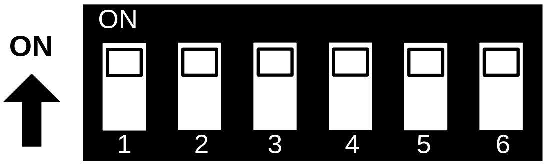
Select whether boot from flash programming through USB device or data flash and NAND flash in CPU.



Select Flash Programming & booting device		
Switch No 1	Switch No 2	Operation description
OFF	OFF	**For Flash Programming** Program firmware to data flash in Eddy-CPU v2.1/V2.5 through USB port.
OFF	ON	Boot from data flash in Eddy-CPU v2.1/v2.5.
ON	OFF	Boot from NAND Flash in Eddy-CPU v2.1/v2.5.
ON	ON	Boot from Data Flash or NAND flash. Data flash connected to SPU will be executed by CPU boot program algorithm when both data flash and NAND flash is boot programmed. If valid ARM vector sequence is undiscovered from data flash, NAND flash boot program will be executed. (refer to datasheet chapter 13 AT91SAM9260 Boot Program)

2.4.2.3 S7:UART Select

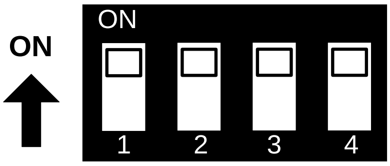
When you set the switch to OFF, UART and serial driver are connected so you can test serial ports.
When you set the switch to ON, each ports will be set to GPIO so that you can check the control status by checking the LED connected to each GPIOs. If you turn on the No. 6 switch, all of 4 ports from the UART will be disconnected from serial driver and GPIO LED but only connected to the expansion header.



Control Serial Port & LED			
Switch Bank	Switch No.	Down Position (OFF) **Serial Port Test**	UP Position (ON) **GPIO TEST (High: LED On)**
S7	1	**UART#0 TEST** TXD, RXD, RTS, CTS signals in UART#0 are connected to RS232 driver.	GPIO (PB4, PB5, PB26, PB27) ports are connected with the GPIO LED of DK board and disconnected with the RS232 Driver.
	2	**UART#0 TEST** DTR, DSR, DCD, RI signal in UART#0 is connected to RS232 driver.	GPIO (PB24, PB22, PB23, PB25) ports are connected with the GPIO LED of DK board and disconnected with the RS232 Driver.
	3	**UART#1 TEST** TXD, RXD, RTS, CTS signals in UART#1 are connected to RS232 driver.	GPIO (PB6, PB7, PB28, PB29) ports are connected with the GPIO LED of DK board and disconnected with the RS232 Driver.
	4	**UART#2 TEST** TXD, RXD, RTS, CTS signals in UART#2 are connected to RS422/485 driver.	GPIO (PB8, PB9, PA4, PA5) ports are connected with the GPIO LED of DK board and disconnected with the RS422/485 Driver.
	5	**UART#3 TEST** TXD, RXD, RTS, CTS signals in UART#3are connected to RS422/485 driver.	GPIO (PB10, PB11, PC8, PC10) ports are connected with the GPIO LED of DK board and disconnected with the RS422/485 Driver.
	6	**For Serial Port & GPIO Test** To test GPIO LED for serial port in DK board, always maintain OFF status.	**Connect to Expansion Header** UART#0~#3 and GPIO LEDs are disconnected with the Eddy-CPU board and directly connected with the Expansion Header (J2, J4).

2.4.2.4 S8:COM3 & S9: COM4 Select

COM Port #3 and COM Port #4 support both RS422/RS485 mode. Switches in the product are used to set configuration for use of RS422/RS485 or echo, non-echo mode for RS485.



COM PORT#3, #4 settings			
Switch Bank	Switch No	Down Position(OFF)	UP Position(ON)
S8 Port#3	1	Set RS485 Half-Duplex	Set RS422 Full-Duplex
	2	RS422(RX enabled) RS485 echo-mode	RS485 non echo-mode
	3	RS422 Termination Resistor not connected	RS422 Termination Resistor Connected
	4	RS485 Termination Resistor not connected	RS422 Termination Resistor Connected
S9 Port#4	1	RS485 Half-Duplex	RS422 Full-Duplex
	2	RS422(RX enabled) RS485 echo-mode	RS485 non echo-mode
	3	RS422 Termination Resistor not connected	RS422 Termination Resistor Connected
	4	RS485 Termination Resistor not connected	RS422 Termination Resistor Connected

2.4.2.5 SW1~SW16: Key Pad

The key pad in DK is designed to use 4 x 4 matrix. When GPIOs are set to the input mode, they will read key values. The key 2, 4, 6, 8 can be used for LCD menu selection. In other words, they can be used as ▲(UP), ▼(DOWN), ◀(LEFT), ▶(RIGHT) direction keys.

GPIOs	Connect with 4 x 4 Key matrix	I/O
PB20	Connect first row	IN
PB21	Connect second row	IN
PB30	Connect third row	IN
PB31	Connect fourth row	IN

PC20	Connect first column	IN
PC21	Connect second column	IN
PC22	Connect third column	IN
PC23	Connect fourth column	IN

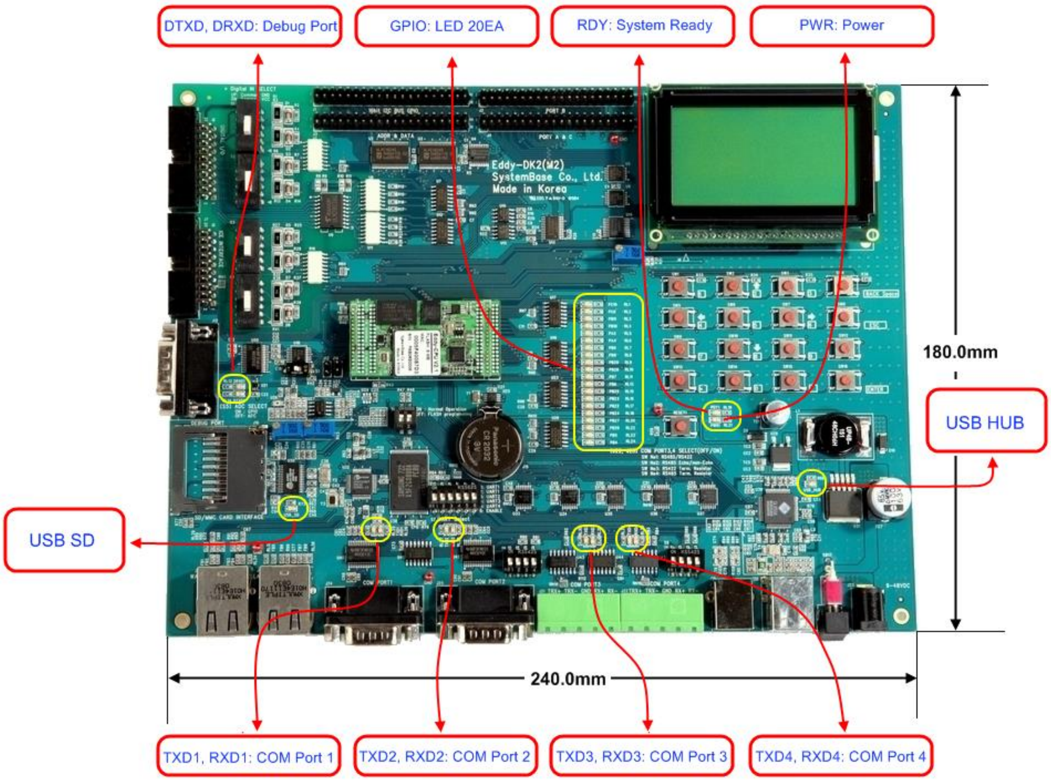
2.4.2.6 SW17: Power

When turned on, this switch will allow power supply.

2.4.2.7 Reset1:Reset

Pin	Function	Description	I/O
PC16	nRESET	Polling Input signal continually from External Reset key, implement as below with checking the constant time of "Low." Less than 5 seconds: General reset function. More than 5 seconds: Factory Default function. When you press and hold reset key while powering up, you can enter boot loader.	IN

2.4.3 LED Description



2.4.3.1 GPIO LED

Eddy-CPU v2.1/v2.5 supports Max 56 GPIO ports. DK board has 20 GPIO LEDs of all GPIO to test. This GPIO LEDs are controlled by UART select switches. (Refer to 2.4.2.3.4 UART Select).

Pin	Function	Description	I/O
PC10	CTS3	UART #3 Clear to Send	I
PC8	RTS3	UART #3 Request to Send	O
PB11	RXD3	UART #3 Receive Data	I
PB10	TXD3	UART #3 Transmit Data	O
PA5	CTS2	UART #2 Cleat to Send	I
PA4	RTS2	UART #2 Request to Send	O
PB9	RXD2	UART #2 Receive Data	I
PB8	TXD2	UART #2 Transmit Data	O
PB29	CTS1	UART #1 Cleat to Send	I
PB28	RTS1	UART #1 Request to Send	O
PB7	RXD1	UART #1 Receive Data	I
PB6	TXD1	UART #1 Transmit Data	O
PB25	RI0	UART #0 Ring Indicator	I
PB23	DCD0	UART #0 Data Carrier Detection	I
PB22	DSR	UART #0 Data Set Ready	O
PB24	DTR0	UART #0 Data Terminal Ready	I
PB27	CTS0	UART #0 Clear to Send	I
PB26	RTS0	UART #0 Request to Send	O
PB5	RXD0	UART #0 Receive Data	I
PB4	TXD0	UART #0 Transmit Data	O

FYI, PIO line has high-drive current capable except PC4-PC31 (2mA), PIO line can drive 16mA. (41.2 DC characteristics in CPU Datasheet, refer to following table)

41.2 DC Characteristics

Symbol	Parameter	Conditions	Min	Type	Max	Units
I_o	Output Current	PA0-PA31 PB0-PB31 PC0-PC3			16	mA
		PC4 - PC31 in 3.3V range			2*	
		PC4 - PC31 in 1.8V range			4	

* Eddy DK v2.1 is 3.3V range so PC4-PC31 PIO can driver 2mA.

2.4.3.2 Power, Ready LED

System Ready (RDY): Indicates that the system is operating normally. (Normal operation: LED blinks)
Power (PWR): Indicates that the power is being supplied. (Red LED ON status)

2.4.3.3 Debug Port LED

DTXD (Debug Port Transmit Data LED): Shows transmission status of the Debug Port.
DRXD (Debug Port Receive Data LED): Shows reception status of the Debug Port.

2.4.3.4 COM Port 1 LED

COM Port 1 Transmit LED: Shows transmission status of COM1 Port.
COM Port 1 Receive LED: Shows reception status of COM1 Port.

2.4.3.5 COM Port 2 LED

COM Port 2 Transmit LED: Shows transmission status of COM2 Port.
COM Port 2 Receive LED: Shows reception status of COM2 Port.

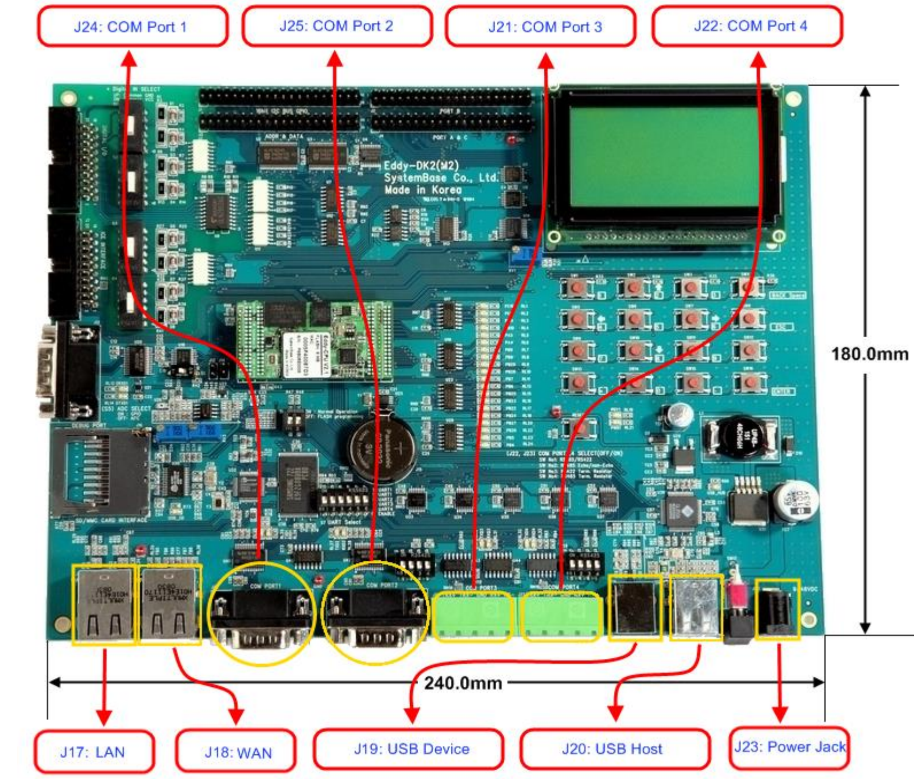
2.4.3.6 COM Port 3 LED

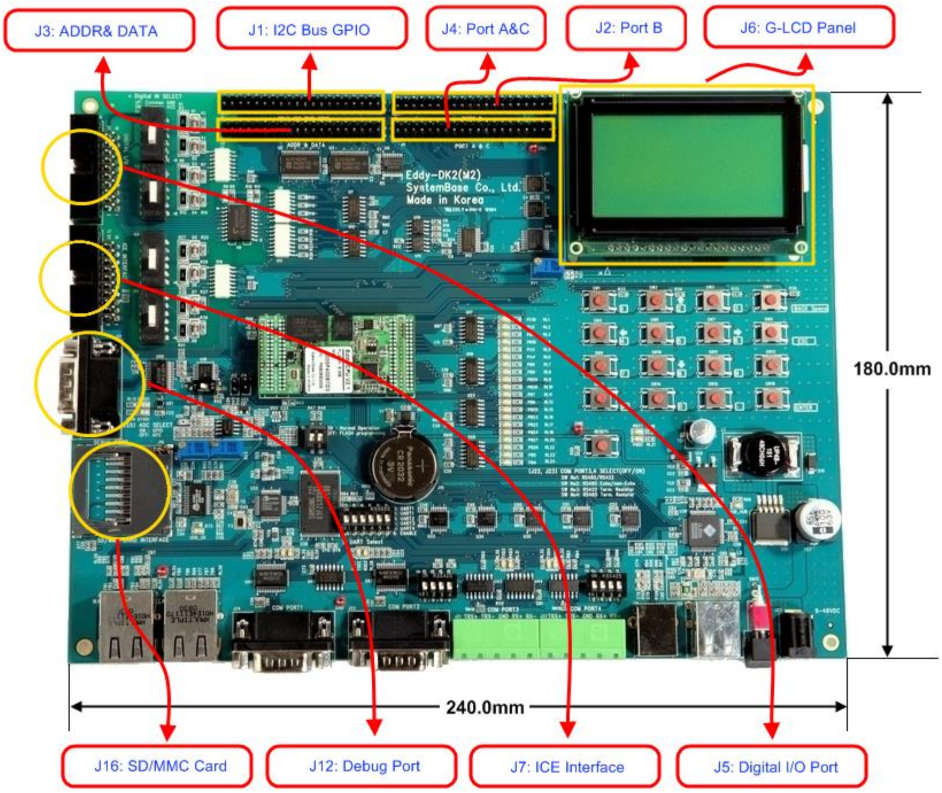
COM Port 3 Transmit LED: Shows transmission status of COM3 Port.
COM Port 3 Receive LED: Shows reception status of COM3 Port.

2.4.3.7 COM Port 4 LED

COM Port 4 Transmit LED: Shows transmission status of COM4 Port.
COM Port 4 Receive LED: Shows reception status of COM4 Port.

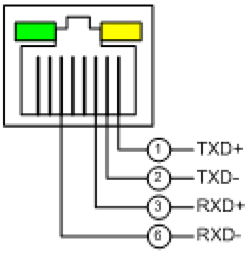
2.4.4 Device Interface





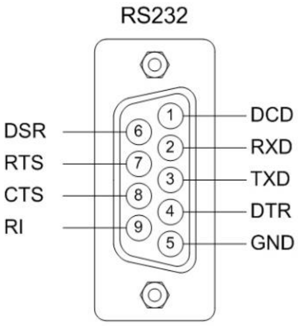
2.4.4.1 WAN and LAN Interface

LAN Port automatically detects Cross/Direct. (Auto MDI/MDIX)



Pin	Signal	Description
1	TXD+	Transmit Data +
2	TXD-	Transmit Data -
3	RXD+	Receive Data +
6	RXD-	Receive Data -
LED		**Description**
Left Green		While 100BaseT Link, it is ON, but while 10BaseT Link it is OFF.
Right Yellow		On when connected to network, blinks when data is received or sent.

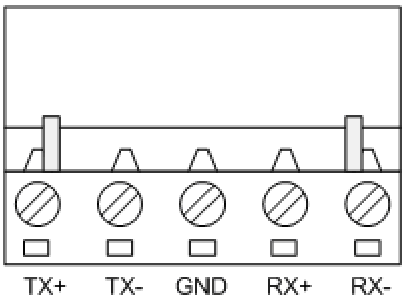
2.4.4.2 COM Port 1 and COM Port 2



RS232

Pin	Signal	Description
1	DCD	Data Carrier Detection (Input) (COM Port 1 only)
2	RXD	Receive Data (Input)
3	TXD	Transmit Data (Output)
4	DTR	Data Terminal Ready (Output) (COM Port 1 only)
5	GND	Ground
6	DSR	Data Set Ready (input) (COM Port 1 only)
7	RTS	Request to Send (Output)
8	CTS	Clear to Send (Input)
9	RI	Ring Indicator (Input)

2.4.4.3 COM Port 3 and COM Port 4



RS422 Full Duplex

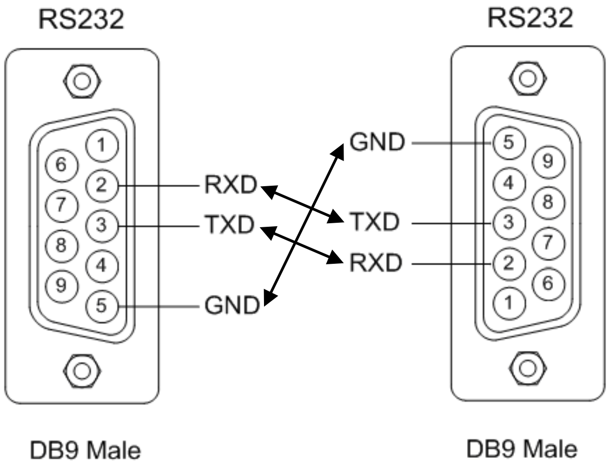
Pin	Signal	Description
1	TXD+	Transmit differential data positive (Output)
2	TXD-	Transmit differential data negative (Output)
3	GND	Ground
4	RXD+	Receive differential data positive (Input)
5	RXD-	Receive differential data negative (Input)

RS485 Half Duplex

Pin	Signal	Description
1	TRX+	Transmit/Receive differential data positive
2	TRX-	Transmit/Receive differential data negative

2.4.4.4 Debug Port

With a debug port, a debug message from a product or status can be checked.



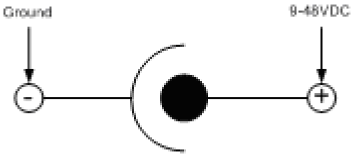
Environment Setting

Debug port is configured as follows so that user has to set his or her PC serial port connected to debug port as follows.

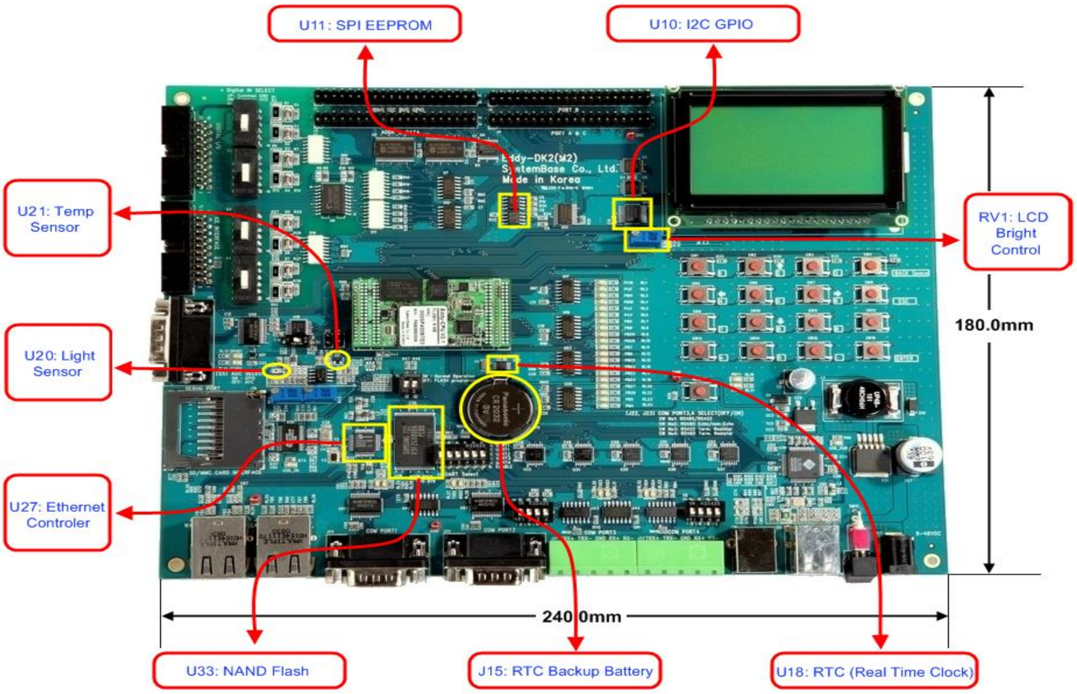
- Speed: 115,200 bps
- Data bit: 8 bit
- Parity bit: Non Parity
- Stop bit: 1 bit
- Flow control: none

2.4.4.5 Power Jack

Contact	Polarity
Center (D : 2mm)	9-48VDC
Outer (D: 6.5mm)	Ground



2.4.5 Internal Device Description



2.4.5.1 EEPROM

Eddy DK v2.1 is equipped with one EEPROM in SPI1.
SPI1: AT25160, 2K x 8bit

2.4.5.2 LCD Module

PowerTIP PG12864LRU-JCNH11Q, a graphic LCD module, is connected to I2C-Bus I/O Expander IC PCA9539.

Signal Name	Function	Description	I/O
P[00:07]	Data bits	Used for data transfer between the CPU and the LCD module.	I/O
P10	/CS1	Chip enable for D2 (Segment 1 to 64)	IN
P11	/CS2	Chip enable for D3 (Segment 65 to 128)	IN
P12	R/W	R/W signal input is used to select the read /write mode High = Read mode, Low = Write mode	IN

P13	D/ $\overline{\text{I}}$	Register selection input
		High = Data register
		Low = Instruction register (for write)
		Busy flag address counter (for read)
			IN

P14	E	Start enable signal to read or write the data	IN

2.4.5.3 16bit I2C Bus GPIO

Connect to I2C interface and usePCA9539 which has expandable 16-bit I/O. In Eddy DK v2.1, Slave address is set to 0x74 and by how you set A1, A0 address input settings, it can be change from 0x74 to 0x77.
16-bit I/O is used as Digital I/O as shown below and can be used as GPIO since it is connected to Expansion Header. When used as GPIO, separate I/O is available.

Function	Description	I/O
P00-P07	DIO Output, consecutively connects to DO [0:7].	OUT
P00	DIO output, DO0 control	
P01	DIO output, DO1 control	
P02	DIO output, DO2 control	
P03	DIO output, DO3 control	
P04	DIO output, DO4 control	
P05	DIO output, DO5 control	
P06	DIO output, DO6 control	
P07	DIO output, DO7 control	
P10-P17	DIO Input, consecutively connects to DI [0:7].	IN
P10	DIO Input, DI0 input	
P11	DIO Input, DI1 input	
P12	DIO Input, DI2 input	
P13	DIO Input, DI3 input	
P14	DIO Input, DI4 input	
P15	DIO Input, DI5 input	
P16	DIO Input, DI6 input	
P17	DIO Input, DI7 input	IN
/INT	Connect to Eddy-CPU PB16	OUT

2.4.5.4 RTC

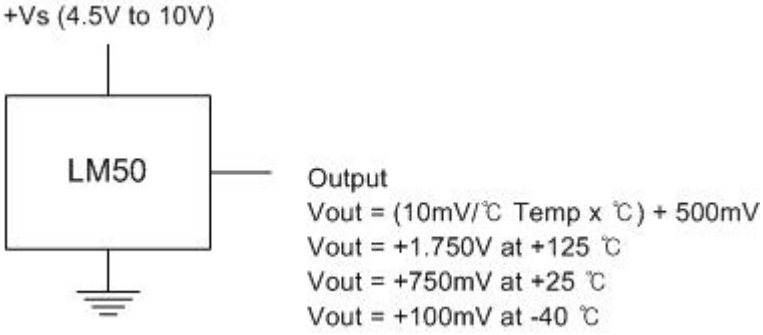
- Use DS1340 connect to I2C interface.
- DS1340 should use crystal which has load capacitance of 12.5pF. (Refer to Crystal Specification below)
- The Crystal specifications are different among RTC Chip check before making a selection.
- CR2032 (235mAh) Lithium battery is used as backup battery.

DS1340 Crystal Specifications

Parameter	Symbol	MIN	TYP	MAX	Units
Normal Frequency	fo		32.768		KHz
Series Resistance	ESR			45,60	KΩ
Load Capacitance	CL		12.5		pF

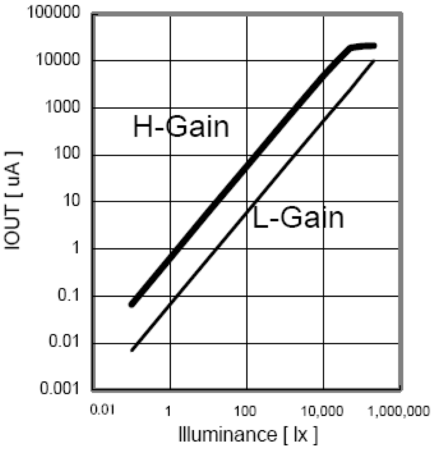
2.4.5.5 Temp Sensor

LM50 by National is used for AD0 (PC0).



2.4.5.6 Light Sensor

BH1600FVC by Rohm is used. Illuminance by different output current is as follows.



The Output voltage is calculated as below
Viout = 0.6 x10-6 x Ev x R1
Where, Viout = IOUT output voltage [V]
Ev = illuminance of the ALS (Ambient Light Sensor) surface [lx]
R1 = IOUT output resistor [Ω]

2.4.5.7 NAND Flash

- 256MB, 8 bit memory (Samsung K9F2G08U0A-PCB0)
- Use of chip Select #3, Address range: 0x4000_0000 ~ 0x4FFF_FFFF.

Eddy-CPU v2.1/V2.5 Signal Name	Function	Description	I/O
A22	CLE	**COMMAND LATCH ENABLE** The CLE input controls the activating path for commands sent to the command register.	OUT
A21	ALE	**ADDRESS LATCH ENABLE** The ALE input controls the activating path for address to the internal address registers.	OUT
NANDOE	NANDOE	data-out control	OUT
NANDWE	NANDWE	controls writes to the I/O port	OUT
PC14(NCS3)	NANDCS	device selection control	OUT
PC17	RDYBSY (R/B)	**READY/BUSY OUTPUT** The R/B output indicates the status of the device operation. When low, it indicates that a program, erase or random read operation is in process and returns to high state upon completion. It is an open drain output and does not float to high-z condition when the chip is deselected or when outputs are disabled.	IN

D[0:7]	DATA bits	**DATA INPUTS/OUTPUTS** The I/O pins are used to input command, address and data, and to output data during read operations. The I/O pins float to high-z when the chip is deselected or when the outputs are disabled.	I/O

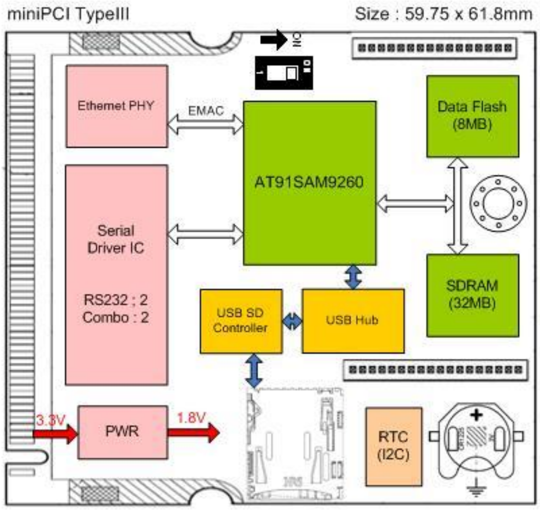
2.4.5.8 Ethernet Controller (WAN Port)

- 16bit mode connection in Davicom DM9000B Ethernet Controller.
- EECS pin, strap option pin internally set as pull-down, should be connected to external pull-up resistor to make the LED to operate.
- Power connected to RJ45 Transformer Center Tap should be connected to DM9000B AVDD18.

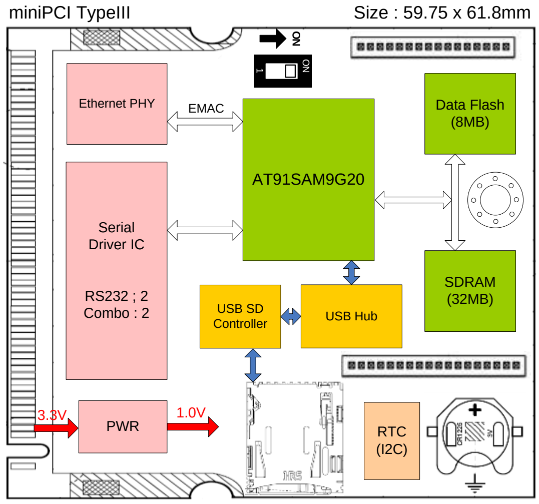
Eddy-CPU v2.1/V2.5 Signal Name	DM9000B Signal Name	Description	I/O
PC12/NCS7	CSN	Chip Select #7 Address : 0x8000 0000-0x8FFF FFFF	OUT
PC15/IRQ1	INTRN	Interrupt polarity depends on the settings in EECK (pin20). 1 : INT pin low active 0 : INT pin high active EECK in DK v2.1 is float and perform as active high.	IN
A2	CMD	**Command Type** When high, Data port When low, INDEX port	OUT
D[0:15]	Data Bus	16-bit mode connection	I/O

2.5 Eddy-S4M v2.1 / v2.5

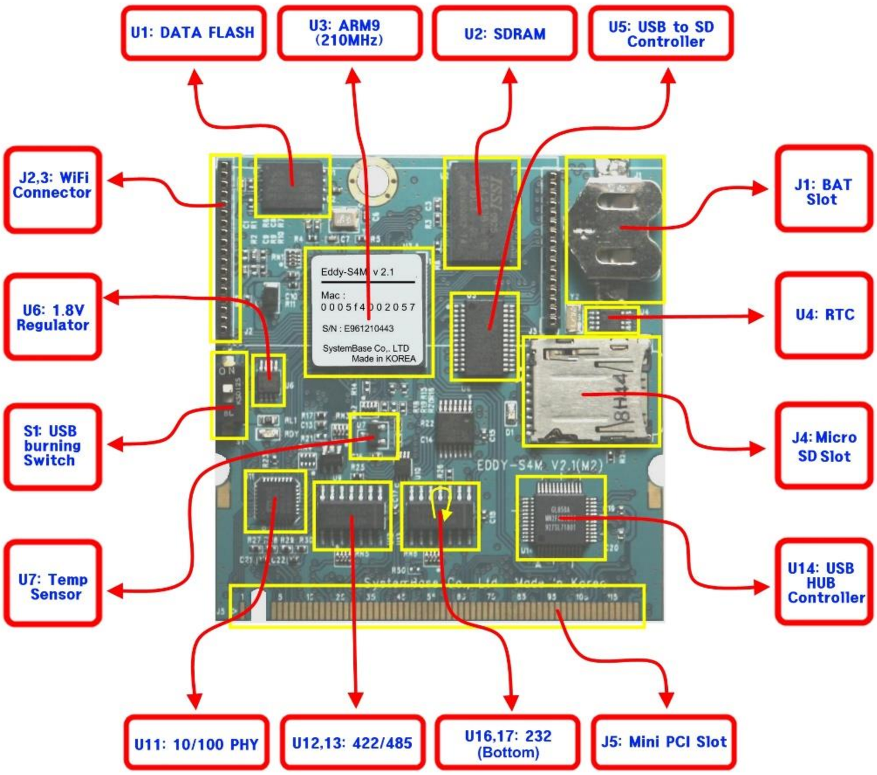
A miniPCI type embedded module with ARM9 processor, 32MB SDRAM, 8MB DataFlash, 10/100Base-T Ethernet port, Max. 34 EA user programmable I/O and MicroSD, RTC, backup battery, 4 port serial (2 x RS232, 2 x Combo). Size of Eddy-S4M is 59.75 x 61.8mm. If it is used with Eddy-S4M-JIG board which is optional, users can develop easily without developing a hardware separately to save time and cost. Additionally, with exemplary source codes and Evaluation Kit circuit, users can build a customized system.
Eddy-S4M v2.1 and Eddy-S4M v2.5 are compatible which uses same DK board and JIG board



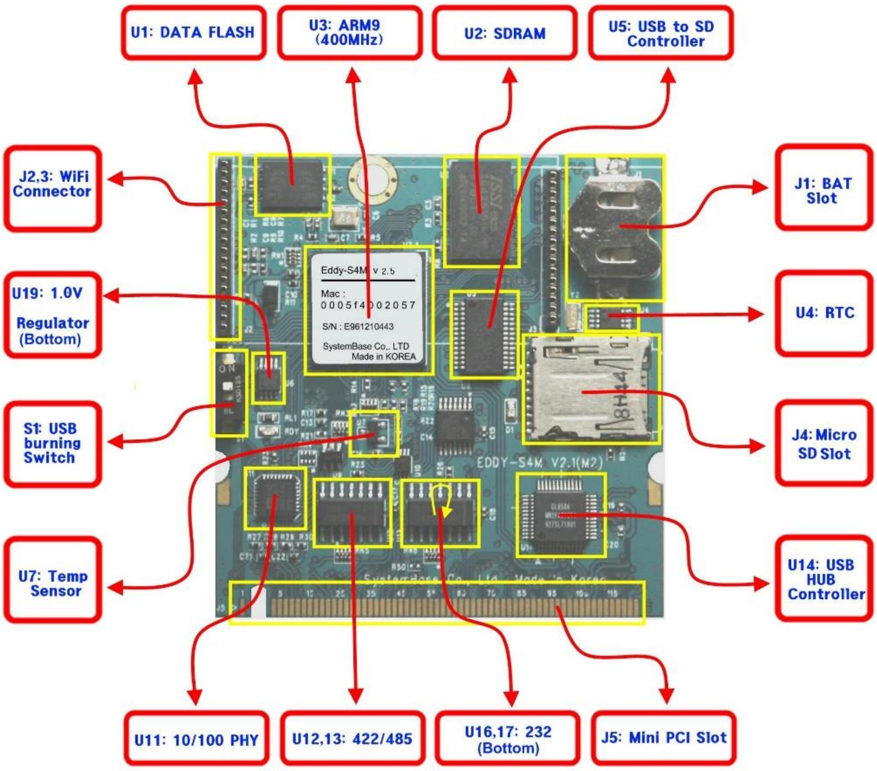
[Eddy-S4M v2.1 Block Diagram]



[Eddy-S4M V2.5 Block Diagram]



[Eddy-S4M v2.1]



[Eddy-S4M v2.5]

2.5.1 miniPCI Card Type III Connector Pinout (J5)

Pin	Signal	Pin	Signal
1	JTAG_TDI	2	JTAG_TDO
	Key		Key
3	JTAG_TMS	4	JTAG_RTCK
5	JTAG_TCK	6	ICE_NTRST
7	LAN_RX+	8	LAN_TX+
9	LAN_RX-	10	LAN_TX-
11	LAN_Speed	12	LAN_LINK
13	P3_RX-	14	RDY#
15	GND	16	NC
17	P3_RX+	18	NC
19	3.3V	20	DCD0
21	P3_TX+	22	DTR0
23	GND	24	3.3V
25	P3_TX-	26	nRESET
27	GND	28	3.3V
29	P4_RX+	30	RxD0#
31	3.3V	32	GND
33	P4_RX-	34	RTS0
35	P4_TX+	36	TxD0#
37	GND	38	CTS0
39	P4_TX-	40	3.3V
41	DEBUG_TxD	42	DSR0
43	DEBUG_RxD	44	RI0
45	PA5	46	RxD1#
47	PA22	48	RTS1
49	GND	50	GND
51	PA30	52	TxD1#
53	NC	54	CTS1
55	GND	56	NRST
57	PB0	58	PB1
59	PB2	60	PB3
61	PB12	62	GND
63	3.3V	64	PB13
65	PB16	66	PB17
67	PB18	68	PB19
69	GND	70	3.3V
71	PB20	72	PB21
73	PB30	74	GND
75	PC0	76	PB31
77	GND	78	PC1
79	PC2	80	PC3
81	PC5	82	GND
83	GND	84	PC9
85	PC10	86	PC12
87	PC13	88	3.3V
89	3.3V	90	PC14
91	PC15	92	PC17
93	PC18	94	PC19
95	PC24	96	PC20
97	NC	98	PC25
99	I2C_TWCK	100	I2C_TWD
101	GND	102	GND
103	DDM	104	DDP
105	DM2	106	DP2
107	DM3	108	DP3
109	DM4	110	DP4
111	SDDATA0	112	SDDATA1
113	SDDATA2	114	GND
115	SDCMD	116	SDDATA3
117	SDCDN	118	SDCLK
119	JTAG_SEL	120	SDWP
121	NC	122	BMS
123	NC	124	3.3V

2.5.2 Board-specific connector pinout

2.5.2.1 ICE and JTAG

S4M Pin No. (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description
1	TDI	-	-	Test Data IN
2	TDO	-	-	Test Data Out
3	TMS	-	-	Test Mode Select
4	RTCK	-	-	Return Test Clock
5	TCK	-	-	Test Clock
6	NTRST	-	-	Test Reset
119	JTAGSEL	-	-	JTAG boundary scan can be used by connecting J3. This pin should not be connected when using ICE (In-Circuit Emulator) or in normal operation status.

2.5.2.2 Ethernet signal from or to PHYceiver

<table>
<tr><th>S4M
Pin No.
(124)</th><th>Name</th><th>S4M-JIG
Pin HDR
(46 x 2)</th><th>S4M-DK
Pin HDR
(46 x 2)</th><th>Description</th></tr>
<tr><td>7</td><td>LAN_RX+</td><td>J5 pin2</td><td>J7 pin2</td><td>Internal CPU Ethernet PHY Physical receive or transmit signal (+ differential)</td></tr>
<tr><td>8</td><td>LAN_TX+</td><td>J5 pin1</td><td>J7 pin1</td><td>Internal CPU Ethernet PHY Physical receive or transmit signal (- differential)</td></tr>
<tr><td>9</td><td>LAN_RX-</td><td>J5 pin3</td><td>J7 pin3</td><td>Internal CPU Ethernet PHY Physical receive or transmit signal (+ differential)</td></tr>
<tr><td>10</td><td>LAN_TX-</td><td>J5 pin4</td><td>J7 pin4</td><td>Internal CPU Ethernet PHY Physical receive or transmit signal (- differential)</td></tr>
<tr><td>11</td><td>LAN_Speed</td><td>J5 pin6</td><td>J7 pin6</td><td>LAN connection status LED
<table>
<tr><th>Link/Activity</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>No Link</td><td>H</td><td>OFF</td></tr>
<tr><td>Link</td><td>L</td><td>ON</td></tr>
<tr><td>Activity</td><td>Toggle</td><td>Blinking</td></tr>
</table></td></tr>
<tr><td>12</td><td>LAN_Link</td><td>J5 pin5</td><td>J7 pin5</td><td>
<table>
<tr><th>Link/Activity</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>No Link</td><td>H</td><td>OFF</td></tr>
<tr><td>Link</td><td>L</td><td>ON</td></tr>
<tr><td>Activity</td><td>Toggle</td><td>Blinking</td></tr>
</table></td></tr>
</table>

2.5.2.3 Serial (RS232 & COMBO) and PIOA (Peripheral I/O Controller A)

S4M Pin No (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description
13	P2_RX-	J4 pin20	J6 pin20	COM port #3 Receive differential data negative (Input) Eddy-S4M built-in RS422/485 inverting receiver input
14	RDY#	J4 pin45	J6 pin45	Indicate CPU activity (Normal operation: Blinks)
17	P2_RX+	J4 pin19	J6 pin19	COM port #3 Receive differential data positive (Input) Eddy-S4M built-in RS422/485 Non-inverting receiver input
20	DCD0	J4 pin9	J6 pin9	COM port #1 Data Carrier Detection signal Eddy-S4M built-in RS232 receiver input
21	P2_TX+	J4 pin17	J6 pin17	COM port #3 Transmit differential data positive (Output) Eddy-S4M built-in RS422/485 Non-inverting driver output
22	DTR0	J4 pin7	J6 pin7	COM port #1 Data Terminal Ready signal Eddy-S4M built-in RS232 driver output
25	P2_TX-	J4 pin18	J6 pin18	COM port #3 Transmit differential data negative (Output) Eddy-S4M built-in RS422/485 inverting driver output
26	nRESET	J4 pin46	J6 pin46	Polling Input signal continually from External Reset key, implement as below with checking the constant time of "Low." Less than 5 seconds: General reset function. More than 5 seconds: Factory Default function.
29	P3_RX+	J4 pin23	J6 pin23	COM port #4 Receive differential data negative (Input) Eddy-S4M built-in RS422/485 Non-inverting receiver input
30	RxD0#	J4 pin4	J6 pin4	COM port #1 Receive Data signal Eddy-S4M built-in RS232 receiver input
33	P3_RX-	J4 pin24	J6 pin24	COM port #4 Receive differential data negative (Input) Eddy-S4M built-in RS422/485 inverting receiver input
34	RTS0	J4 pin5	J6 pin5	COM port #1 Request To Send signal Eddy-S4M built-in RS232 driver output
35	P3_TX+	J4 pin21	J6 pin21	COM port #4 Transmit differential data positive (Output) Eddy-S4M built-in RS422/485 Non-inverting driver output
36	TxD0#	J4 pin3	J6 pin3	COM port #1 Transmit Data signal Eddy-S4M built-in RS232 driver output
38	CTS0	J4 pin6	J6 pin6	COM port #1 Request to Send signal Eddy-S4M built-in RS232 receiver input
39	P3_TX-	J4 pin22	J6 pin22	COM port #4 Transmit differential data negative(Output) Eddy-S4M built-in RS422/485 inverting driver output
41	DTxD#	J4 pin1	J6 pin1	Transmit Data signal of Debug Port Eddy-S4M built-in RS232 driver output
42	DSR0	J4 pin8	J6 pin8	COM port #1 Data Set Ready signal Eddy-S4M built-in RS232 receiver input
43	DRxD	J4 pin2	J6 pin2	Receive Data signal of Debug Port Eddy-S4M built-in RS232 receiver input
44	RI0	J4 pin8	J6 pin8	COM port #1 Ring Indicator signal Eddy-S4M built-in RS232 receiver input
45	PA5	J5 pin7	J7 pin7	Can be used for GPIO only
46	RxD1#	J4 pin12	J6 pin12	COM port #1 Receive Data signal Eddy-S4M built-in RS232 receiver input
47	PA22	J5 pin8	J7 pin8	Can be used for GPIO
48	RTS1	J4 pin13	J6 pin13	COM port #1 Request to Send signal Eddy-S4M built-in RS232 driver output
51	PA30	J5 pin9	J7 pin9	Can be used for GPIO only

52	TxD1#	J4 pin11	J6 pin11	COM port #1 Request to Send signal Eddy-S4M built-in RS232 driver output
54	CTS1	J4 pin14	J6 pin14	COM port #1 Request to Send signal Eddy-S4M built-in RS232 receiver input
56	NRST	J5 pin46	J7 pin46	External device Reset output signal (active low)

2.5.2.4 PIOB and PIOC (Peripheral I/O Controller B/C)

S4M Pin No (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description	
57	PB0	J5 pin11	J7 pin11	Peripheral A : SPI1_MISO	SPI1(Serial Peripheral Interface) Master In Slave Out
				Peripheral B : TIOA3	Timer Counter ch3 I/O Line A
58	PB1	J5 pin12	J7 pin12	Peripheral A : SPI1_MOSI	SPI1(Serial Peripheral Interface) Master Out Slave In
				Peripheral B : TIOB3	Timer Counter ch3 I/O Line B
59	PB2	J5 pin13	J7 pin13	Peripheral A : SPI1_SPCK	SPI1(Serial Peripheral Interface) Serial Clock
60	PB3	J5 pin14	J7 pin14	Peripheral A : SPI1_NPCS0	SPI1(Serial Peripheral Interface) Peripheral Chip Select 0
				Peripheral B : TIOA5	Timer Counter ch5 I/O Line A
61	PB12	J5 pin17	J7 pin17	Peripheral A : TXD5	USART5 Transmit Data
64	PB13	J5 pin18	J7 pin18	Peripheral A : RXD5	USART5 Receive Data
65	PB16	J5 pin119	J7 pin119	Peripheral A : TK0	SSC Transmit Clock
				Peripheral B : TCLK3	Timer Counter ch3 External CLK IN
66	PB17	J5 pin20	J7 pin20	Peripheral A : TF0	SSC Transmit Frame Sync
				Peripheral B : TCLK4	Timer Counter ch4 External CLK IN
67	PB18	J5 pin21	J7 pin21	Peripheral A : TD0	SSC Transmit Data
				Peripheral B : TIOB4	Timer Counter ch4 I/O Line B
68	PB19	J5 pin22	J7 pin22	Peripheral A : RD0	SSC Receive Data
				Peripheral B : TIOB5	Timer Counter ch5 I/O Line B
71	PB20	J5 pin23	J7 pin23	Peripheral A : RK0	SSC Receive Clock

72	PB21	J5 pin24	J7 pin24	Peripheral A : RF0	SSC Receive Frame Sync
73	PB30	J5 pin25	J7 pin25	Peripheral A : PCK0	Programmable Clock Output 0
75	PC0	J5 pin27	J7 pin27	Peripheral A : AD0	Analog to Digital Converter Input Ch0
76	PB31	J5 pin26	J7 pin26	Peripheral A : PCK1	Programmable Clock Output 1
78	PC1	J5 pin28	J7 pin28	Peripheral A : AD1	Analog to Digital Converter Input Ch1
				Peripheral B : PCK0	Programmable Clock Output 0
79	PC2	J5 pin29	J7 pin29	Peripheral A : AD2	Analog to Digital Converter Input Ch2
				Peripheral B : PCK1	Programmable Clock Output 1
80	PC3	J5 pin30	J7 pin30	Peripheral A : AD3	Analog to Digital Converter Input Ch3
				Peripheral B : SPI1_NPCS3	SPI1(Serial Peripheral Interface) Peripheral Chip Select 3
81	PC5	J5 pin33	J7 pin33	Peripheral B : SPI1_NPCS1	SPI1(Serial Peripheral Interface) Peripheral Chip Select 1
84	PC9	J5 pin34	J7 pin34	Can be used for GPIO only	
85	PC10	J5 pin35	J7 pin35	Can be used for GPIO only	
86	PC12	J5 pin36	J7 pin36	Can be used for GPIO only	
87	PC13	J5 pin37	J7 pin37	Can be used for GPIO only	
90	PC14	J5 pin38	J7 pin38	Can be used for GPIO only	
91	PC15	J5 pin39	J7 pin39	Can be used for GPIO only	
92	PC17	J5 pin40	J7 pin40	Can be used for GPIO only	
93	PC18	J5 pin41	J7 pin41	Peripheral B : SPI1_NPCS1	SPI1(Serial Peripheral Interface) Peripheral Chip Select 1
94	PC19	J5 pin42	J7 pin42	Peripheral B : SPI1_NPCS2	SPI1(Serial Peripheral Interface) Peripheral Chip Select 2
95	PC24	J5 pin44	J7 pin44	Can be used for GPIO only	
96	PC20	J5 pin43	J7 pin43	Peripheral B : SPI1_NPCS3	SPI1(Serial Peripheral Interface) Peripheral Chip Select 3
98	PC25	J5 pin45	J7 pin45	Can be used for GPIO only	

2.5.2.5 Two Wire Interface

S4M Pin No (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description
99	I2C_TWCK	J4 pin43	J6 pin43	Two-wire Serial Clock. This can be used GPIO pin unless RTC function is used.
100	I2C_TWD	J4 pin44	J6 pin44	Two-wire Serial Clock. This can be used GPIO pin unless RTC function is used.

2.5.2.6 Universal Serial Bus

S4M Pin No (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description
103	DDM	J4 pin25	J6 pin25	USB Device Port Data –
104	DDP	J4 pin26	J6 pin26	USB Device Port Data +
105	DM2	J4 pin27	J6 pin27	USB Port2 Data –. Connected to DSPORT2 in GL850A USB 2.0 Hub Controller.
106	DP2	J4 pin27	J6 pin27	USB Port2 Data +. Connected to DSPORT2 in GL850A USB 2.0 Hub Controller.
107	DM3	J4 pin29	J6 pin29	USB Port3 Data –. Connected to DSPORT3 in GL850A USB 2.0 Hub Controller.
108	DP3	J4 pin30	J6 pin30	USB Port3 Data +. Connected to DSPORT3 in GL850A USB 2.0 Hub Controller.
109	DM4	J4 pin33	J6 pin33	USB Port4 Data -. Connected to DSPORT4 in GL850A USB 2.0 Hub Controller.
110	DP4	J4 pin34	J6 pin34	USB Port4 Data +. Connected to DSPORT4 in GL850A USB 2.0 Hub Controller.

2.5.2.7 Multimedia Card Interface

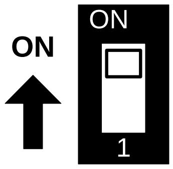
S4M Pin No (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description
111	SDDATA0	J4 pin35	J6 pin35	SD Data0
112	SDDATA1	J4 pin36	J6 pin36	SD Data1
113	SDDATA2	J4 pin37	J6 pin37	SD Data2
115	SDCMD	J4 pin38	J6 pin38	SD command
116	SDDATA3	J4 pin39	J6 pin39	SD Data3
117	SDCDN	J4 pin40	J6 pin40	SD card detect
118	SDCLK	J4 pin41	J6 pin41	SD Clock
120	SDWP	J4 pin42	J6 pin42	SD Write Protect

122	BMS	-	-	Boot Mode Select signal
				BMS = 1, Boot on Embedded ROM
				BMS = 0, Boot on External Memory

2.5.2.8 Etc.

S4M Pin No (124)	Name	S4M-JIG Pin HDR (46 x 2)	S4M-DK Pin HDR (46 x 2)	Description
16, 18, 53, 97, 121, 123	NC	J5 pin10	J5 pin10	No Connection
15, 23, 27, 32, 37, 49, 50, 55, 62, 69, 74, 77, 82, 83, 101, 102, 114	GND	J4: 31,32 J5: 31,32	J6: 31,32 J7: 31,32	Ground
19, 24, 28, 31, 40, 63, 70, 88, 89, 124	3.3V	J4: 15,16	J6: 15,16	3.0 to 3.6V power input

2.5.3 Switch operation



Switch No 1	Operation description
OFF	**For Flash Programming** Through USB device port, firmware images are saved into the Flash memory (Only available by Windows host). For detailed information, refer to chapter 9 System recovery in this manual.
ON	Boot through Eddy-S4M v2.1 data flash.

2.5.4 LED operation

System Ready (RDY) indicated that system is operating normally. (Normal operation: Blinks)

2.5.5 Ethernet

Eddy-S4M has built-in KSZ8041NL PHY so that RJ45 connector with built-in transformer can be connected to implement Ethernet.

WARNING: RJ45 with built-in transformer maybe different among products. Therefore, when designing a board, check your pin numbers for internal circuit for RJ45 connector.

KSZ8041NL features are as follows.

- Fully compliant to IEEE 802.3u Standard
- Supports MDI/MDI-X auto crossover (Auto-MDI)
- MII interface support
- RMII interface support with external 50MHz system clock
- ESD rating (6kV)
- Built-in 1.8V regulator for core
- Available in 32-pin (5 x 5mm) MLF® package

2.5.6 RTC

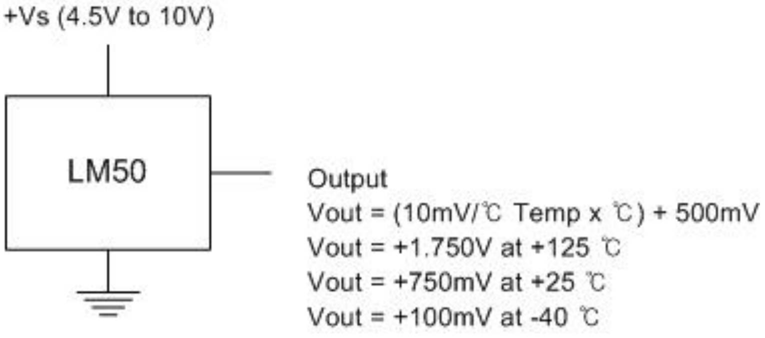
- Use DS1340 connected by I2C interface.
- For DS1340, crystal should be used with load capacitance of 12.5pF. (Refer to Crystal Specification below)
- Crystal specification is different among RTC Chip, check before selecting parts.
- CR2032 (235mAh) Lithium battery is used for a backup battery.

DS1340 Crystal Specifications

Parameter	Symbol	MIN	TYP	MAX	Units
Normal Frequency	fo		32.768		KHz
Series Resistance	ESR			45,60	KΩ
Load Capacitance	CL		12.5		pF

2.5.7 Temp Sensor

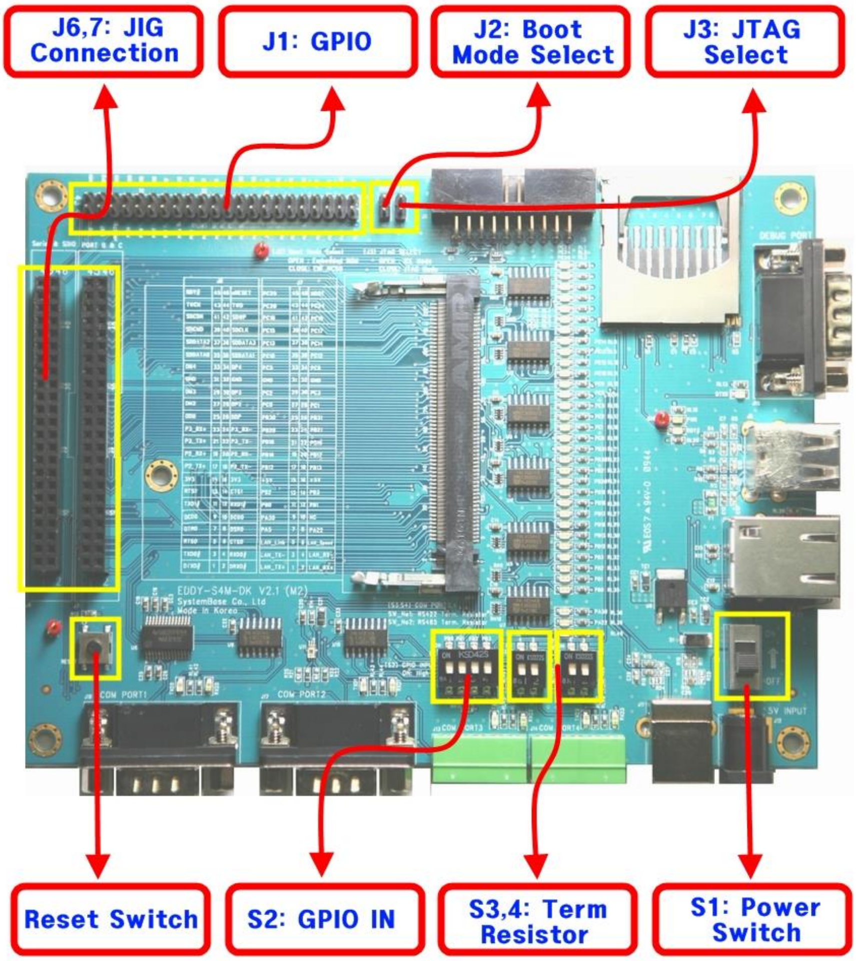
LM50 by National is used in AD0 (PC0).



2.6 Eddy-S4M-DK v2.1

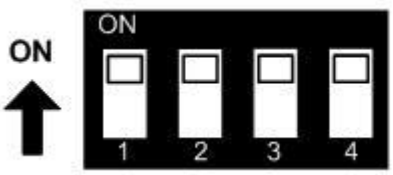
Eddy-S4M development kit (DK) is mounted with Eddy-S4M so that the programmer can easily upload his or her application and test.

2.6.1 Descriptions for switch and connector



2.6.1.1 S2 : GPIO input setting

Set PB0-PB4 as input and configure switches to check if input value is changed.



Switch No		Down Position(OFF)	UP Position(ON)
1	PB0 input value	Low	High
2	PB0 input value	Low	High
3	PB0 input value	Low	High
4	PB0 input value	Low	High

2.6.1.2 S3,4 : select terminal resistor

COM Port #3 and COM Port #4 are Combo ports supporting RS422/RS485. Terminal resistors in these ports are configured by switches located above each terminal block ports.

Switch No	Down Position(OFF)	UP Position(ON)
1	RS422 Termination Resistor not connected	RS422 Termination Resistor Connected
2	RS485 Termination Resistor not connected	RS422 Termination Resistor Connected
1	RS422 Termination Resistor not connected	RS422 Termination Resistor Connected
2	RS485 Termination Resistor not connected	RS422 Termination Resistor Connected

2.6.1.3 J6,J7 : JIG board connector(Socket)

J6

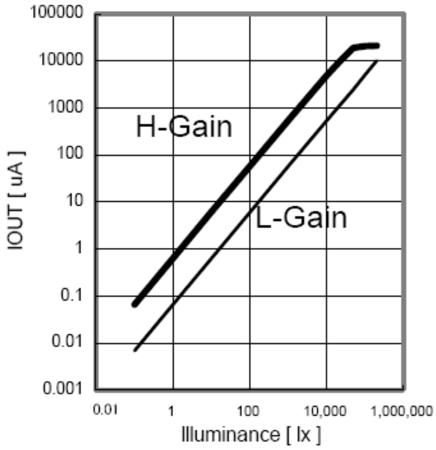
Pin	Signal	Pin	Signal
1	DTxD	2	DRxD
3	TxD0#	4	RxD0#
5	RTS0	6	CTS0
7	DTR0	8	DSR0
9	DCD0	10	RI0
11	TxD1#	12	RxD1#
13	RTS1	14	CTS1
15	3.3V	16	3.3V
17	P3_TX+	18	P3_TX-
19	P3_RX+	20	P3_RX-
21	P4_TX+	22	P4_TX-
23	P4_RX+	24	P4_RX-
25	DDM	26	DDP
27	DM2	28	DP2
29	DM3	30	DP3
31	GND	32	GND
33	DM4	34	DP4
35	SDDATA0	36	SDDATA1
37	SDDATA2	38	SDDATA3
39	SDCMD	40	SDCLK
41	SDCDN	42	SDWP
43	TWCK	44	TWD
45	RDY#	46	nRESET(IN)

J7

Pin	Signal	Pin	Signal
1	LAN_TX+	2	LAN_RX+
3	LAN_TX -	4	LAN_RX-
5	LAN_LINK	6	LAN_Speed
7	PA5	8	PA22
9	PA30	10	NC
11	PB0	12	PB1
13	PB2	14	PB3
15	5V	16	5V
17	PB12	18	PB13
19	PB16	20	PB17
21	PB18	22	PB19
23	PB20	24	PB21
25	PB30	26	PB31
27	PC0	28	PC1
29	PC2	30	PC3
31	GND	32	GND
33	PC5	34	PC9
35	PC10	36	PC12
37	PC13	38	PC14
39	PC15	40	PC17
41	PC18	42	PC19
43	PC20	44	PC24
45	PC25	46	NRST(OUT)

2.6.1.4 U7 : Light Sensor

BH1600FVC by Rohm is used. Illuminance by different output current is as follows.



The Output voltage is calculated as below

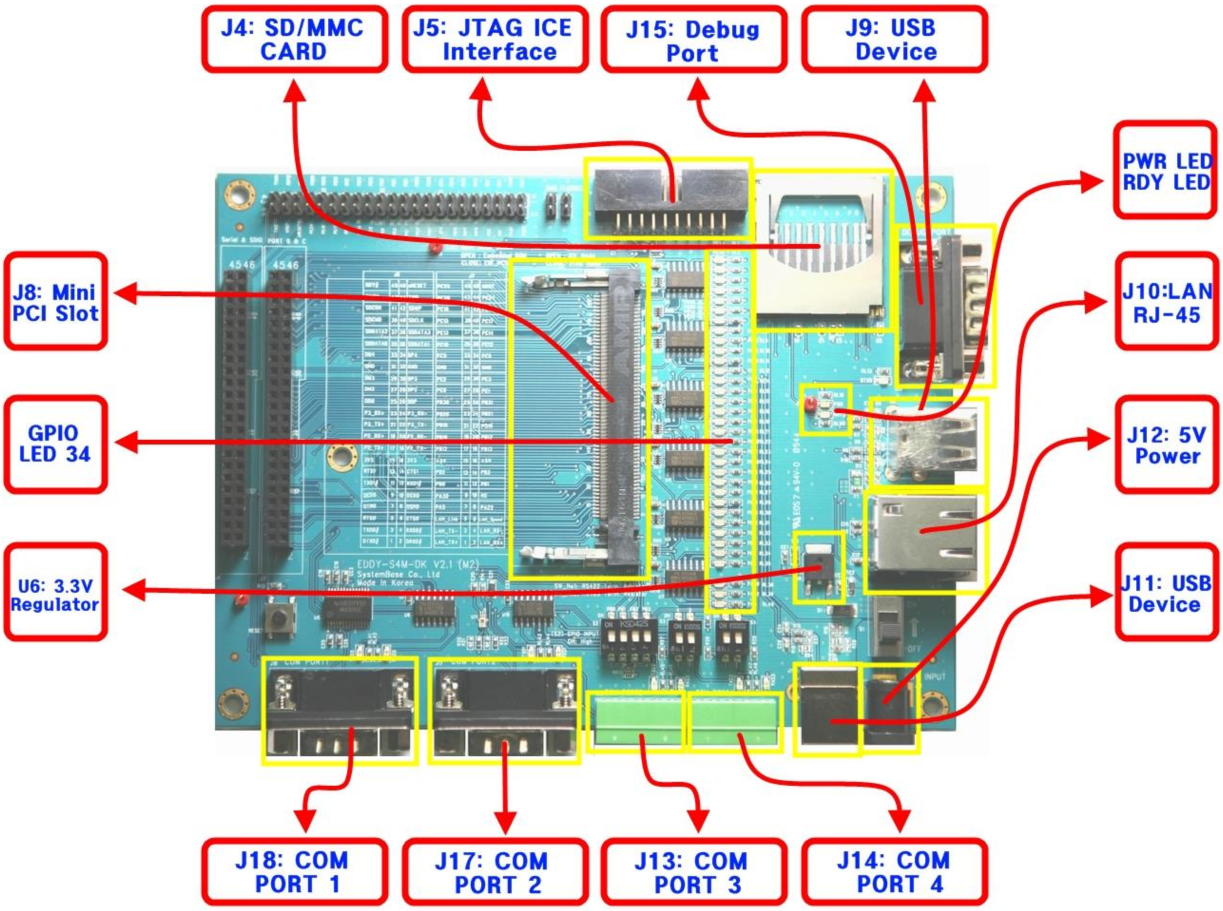
Viout = 0.6 x10-6 x Ev x R1

Where, Viout = IOUT output voltage [V]

Ev = illuminance of the ALS(Ambient Light Sensor) surface [lx]

R1 = IOUT output resistor [Ω]

2.6.2 Interface



2.6.2.1 Power, Ready LED

System Ready (RDY): Indicates system is operating normally. (Normal operation: Blinking)
Power (PWR): Indicates power is supplied. (Red LED: ON status)

2.6.2.2 Serial Port LED

Pin name	Function	Description
Debug Port	TxD	Debug Port displays receiving/transmitting status
	RxD	Debug Port displays receiving status
COM Port 1 (RS232)	TxD	COM Port1 displays transmitting status
	RxD	COM Port1 displays receiving status
COM Port 2 (RS232)	TxD	COM Port2 displays transmitting status
	RxD	COM Port2 displays receiving status
COM Port 3 (RS422/RS485)	TxD	For RS422, COM Port3 displays transmitting status For RS485, displays receiving/transmitting status
	RxD	For RS422, COM Port3 displays receiving status For RS485, N/A (LED off)
COM Port 4 (RS422/RS485)	TxD	For RS422, COM Port4 displays transmitting status For RS485, displays receiving/transmitting status
	RxD	For RS422, COM Port4 displays receiving status For RS485, N/A (LED off)

2.6.2.3 GPIO LED

Eddy-S4M provides total 34 GPIOs.

No	Pin name	Description	I/O
1	PC25	Can be used for GPIO only	I/O
2	PC24	Can be used for GPIO only	I/O
3	PC20	GPIO or SPI1_NPCS3	I/O
4	PC19	GPIO or SPI1_NPCS2	I/O
5	PC18	GPIO or SPI1_NPCS1	I/O
6	PC17	Can be used for GPIO only	I/O
7	PC15	Can be used for GPIO only	I/O
8	PC14	Can be used for GPIO only	I/O
9	PC13	Can be used for GPIO only	I/O
10	PC12	Can be used for GPIO only	I/O
11	PC10	Can be used for GPIO only	I/O
12	PC9	Can be used for GPIO only	I/O
13	PC5	GPIO or SPI1_NPCS1	I/O
14	PC3	GPIO or AD3 or SPI1_NPCS3	I/O
15	PC2	GPIO or AD2 or PCK0	I/O
16	PC1	GPIO or AD1 or PCK0	I/O
17	PC0	GPIO or AD0	I/O
18	PB31	GPIO or PCK1	I/O
19	PB30	GPIO or PCK0	I/O
20	PB21	GPIO or RF0	I/O
21	PB20	GPIO or RK0	I/O

22	PB19	GPIO or RTD0 or TIOB5	I/O
23	PB18	GPIO or TD0 or TIOB4	I/O
24	PB17	GPIO or TF0 or TCLK4	I/O
25	PB16	GPIO or RxD5 or TCLK3	I/O
26	PB13	GPIO or RxD5	I/O
27	PB12	GPIO or TxD5	I/O
28	PB3	GPIO or SPI1_NPCS0 or TIOA5	I/O
29	PB2	GPIO or SPI1_SPCK	I/O
30	PB1	GPIO or SPI1_MOSI or TIOB3	I/O
31	PB0	GPIO or SPI1_MISO or TIOA3	I/O
32	PA30	Can be used for GPIO only	I/O
33	PA22	Can be used for GPIO only	I/O
34	PA5	Can be used for GPIO only	I/O

FYI, PIO line has high-drive current capable so except PC4-PC31 (2mA) PIO line can driver 16mA. (41.2 DC characteristics from CPU Datasheet, refer to table below)

AT91SAM9260 DC Characteristics

Symbol	Parameter	Conditions	Min	Type	Max	Units
I_o	Output Current	PA0-PA31 PB0-PB31 PC0-PC3			16	mA
		PC4 - PC31 in 3.3V range			2*	
		PC4 - PC31 in 1.8V range			4	

* Eddy DK v2.1 has 3.3V range so that PC4-PC31 PIO can driver 2mA.

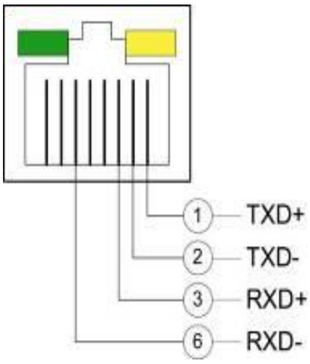
2.6.2.4 J10 : Ethernet

Eddy-S4M has built-in KSZ8041NL PHY so that RJ45 connector with built-in transformer can be connected to implement Ethernet.

WARNING: RJ45 with built-in transformer maybe different among products. Therefore, when designing a board, check your pin numbers for internal circuit for RJ45 connector.

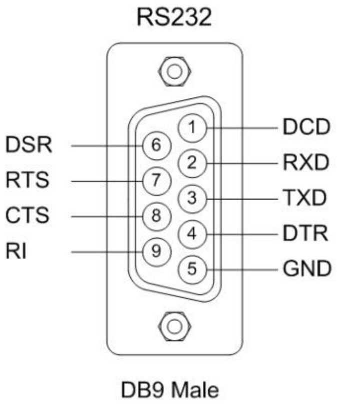
Features of KSZ8041NL are as follows.

- Fully compliant to IEEE 802.3u Standard
- Supports MDI/MDI-X auto crossover (Auto-MDI)
- MII interface support
- RMII interface support with external 50MHz system clock
- ESD rating (6kV)
- Built-in 1.8V regulator for core
- Available in 32-pin (5mm x 5mm) MLF® package



<table>
<tr><th>Pin</th><th>Signal</th><th>Description</th></tr>
<tr><td>1</td><td>TXD+</td><td>Physical transmit or receive signal (+ differential)</td></tr>
<tr><td>2</td><td>TXD-</td><td>Physical transmit or receive signal (- differential)</td></tr>
<tr><td>3</td><td>RXD+</td><td>Physical transmit or receive signal (+ differential)</td></tr>
<tr><td>6</td><td>RXD-</td><td>Physical transmit or receive signal (- differential)</td></tr>
<tr><th colspan="2">LED</th><th>Description</th></tr>
<tr><td colspan="2">Left Green</td><td>LAN Connection Speed
<table>
<tr><th>Speed</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>10Base-T</td><td>H</td><td>OFF</td></tr>
<tr><td>100Base-TX</td><td>L</td><td>ON</td></tr>
</table></td></tr>
<tr><td colspan="2">Right Yellow</td><td>LAN Connection Status
<table>
<tr><th>Speed</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>No Link</td><td>H</td><td>OFF</td></tr>
<tr><td>Link</td><td>L</td><td>ON</td></tr>
<tr><td>Activity</td><td>Toggle</td><td>Blinking</td></tr>
</table></td></tr>
</table>

2.6.2.5 J17, 18 : COM Port 1 and Port 2

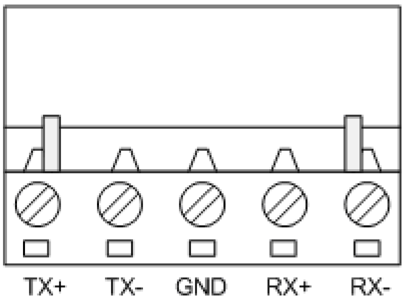


RS232

Pin	Signal	Description
1	DCD	Data Carrier Detection (Input) (COM Port 1 only)
2	RXD	Receive Data (Input)
3	TXD	Transmit Data (Output)
4	DTR	Data Terminal Ready (Output) (COM Port 1 only)
5	GND	Ground
6	DSR	Data Set Ready (input) (COM Port 1 only)
7	RTS	Request to Send (Output)
8	CTS	Clear to Send (Input)
9	RI	Ring Indicator (Input)

* COM Port 2 only provides TxD, RxD, RTS, and CTS signals.

2.6.2.6 J13, 14 : COM Port 3 and Port 4



RS422 Full Duplex

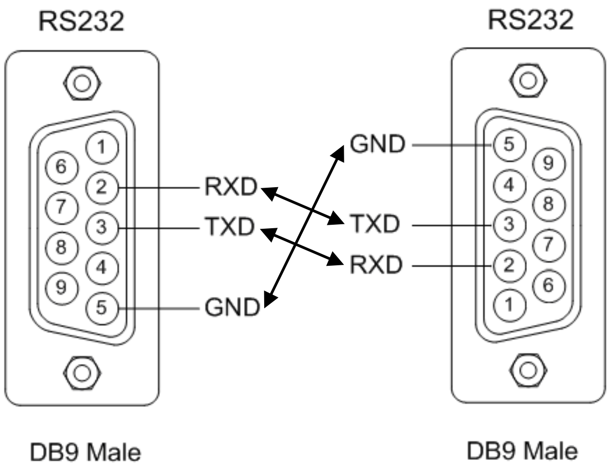
Pin	Signal	Description
1	TXD+	Transmit differential data positive (Output)
2	TXD-	Transmit differential data negative (Output)
3	GND	Ground
4	RXD+	Receive differential data positive (Input)
5	RXD-	Receive differential data negative (input)

RS485 Half Duplex

Pin	Signal	Description
1	TRX+	Transmit/Receive differential data positive
2	TRX-	Transmit/Receive differential data negative

2.6.2.7 J15 : Debug Port

You can check debug message or status information with debug port.

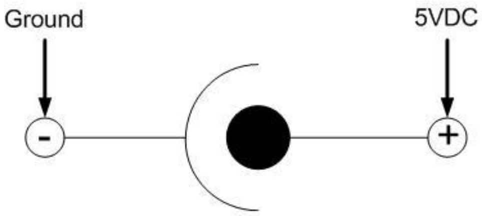


Environment Setting

Debug port is configured as follows so user has to set his or her PC serial port connected to debug port as follows.
- Speed: 115200 bps
- Data bit: 8 bit
- Parity bit: Non Parity
- Stop bit: 1 bit
- Flow control: none

2.6.2.8 S1 : Power Jack

Contact	Polarity
Center (D : 2mm)	5VDC
Outer (D: 6.5mm)	Ground

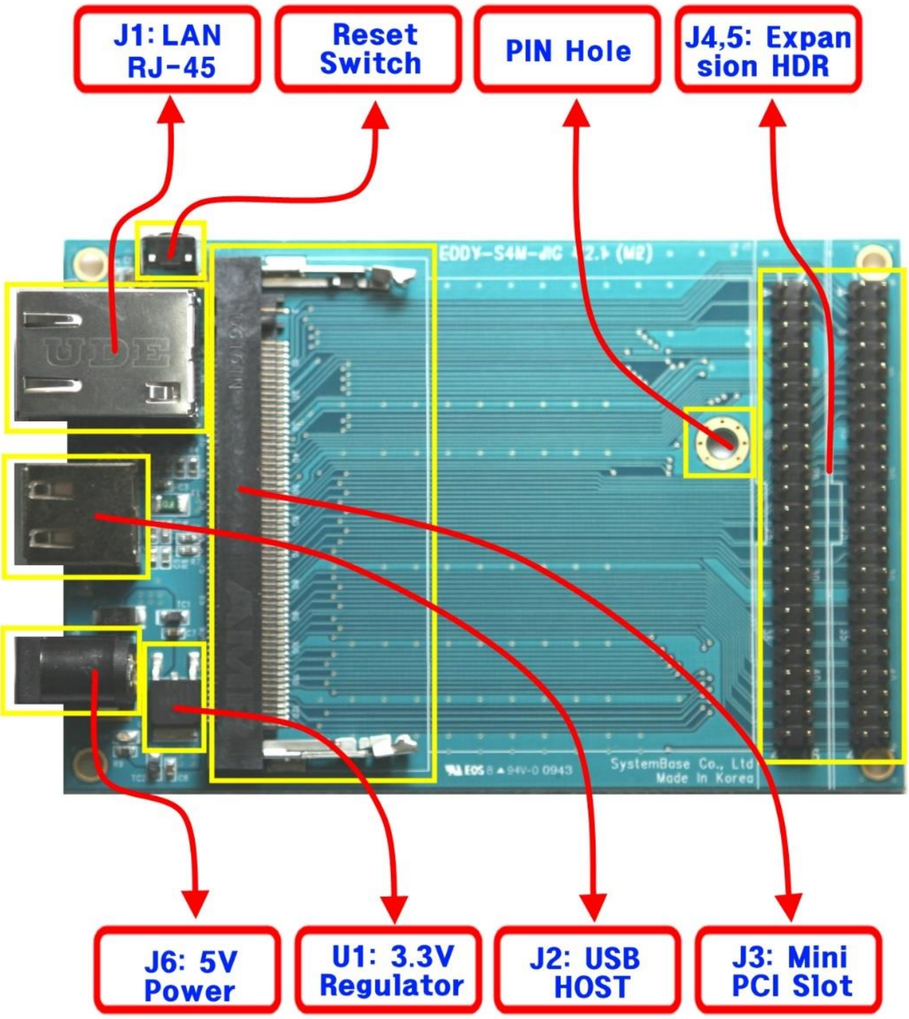


GPIO Connector pinout

Pin	Signal	Pin	Signal
1	PA5	2	PA22
3	PA30	4	NC
5	PB0	6	PB1
7	PB2	8	PB3
9	PB12	10	PB13
11	PB16	12	PB17
13	PB18	14	PB19
15	3.3V	16	3.3V
17	PB20	18	PB21
19	PB30	20	PB31
21	PC0	22	PC1
23	PC2	24	PC3
25	PC5	26	PC9
27	PC10	28	PC12
29	PC13	30	PC14
31	GND	32	GND
33	PC15	34	PC17
35	PC18	36	PC19
37	PC20	38	PC24
39	PC25	40	nRESET (IN)
41	RDY#	42	NRST (OUT)
43	TWCK	44	TWD

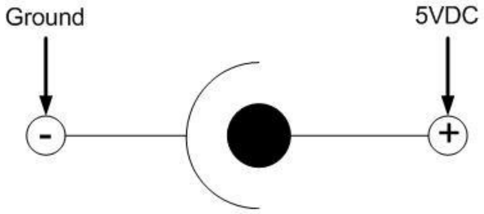
2.7 Eddy-S4M-JiG v2.1

Eddy-S4M JIG board is a test board which enables for the user to integrate and test their application with Eddy-S4M. JIG board including mini connector for joining Eddy-S4M, Ethernet RJ45, USB Host, Power, Reset Switch, and providing connectors to all Eddy-S4M functions.



2.7.1 J6 : Power Jack

Contact	Polarity
Center (D : 2mm)	5 VDC
Outer (D: 6.5mm)	Ground



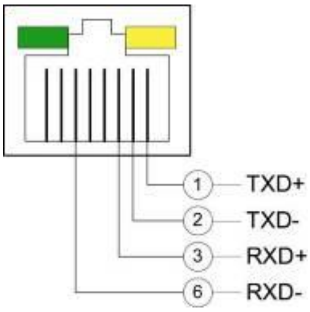
2.7.2 J1 : Ethernet

Since there is KSZ8041NL PHY in Eddy-S4M module, when integrating Ethernet, just connect RJ45 where transformer is located

WARNING: When you use RJ45 which has transformer in its internal circuit, it is possible to each product doesn’t have equal PIN spec. Therefore, you must confirm PIN number

Below is KSZ8041NL functions

- Fully compliant to IEEE 802.3u Standard
- Supports MDI/MDI-X auto crossover (Auto-MDI)
- MII interface support
- RMII interface support with external 50MHz system clock
- ESD rating (6kV)
- Built-in 1.8V regulator for core
- Available in 32-pin (5mm x 5mm) MLF® package

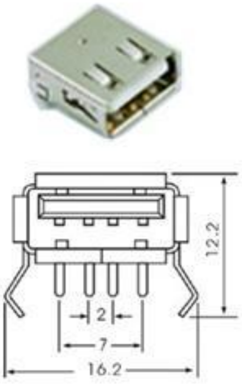


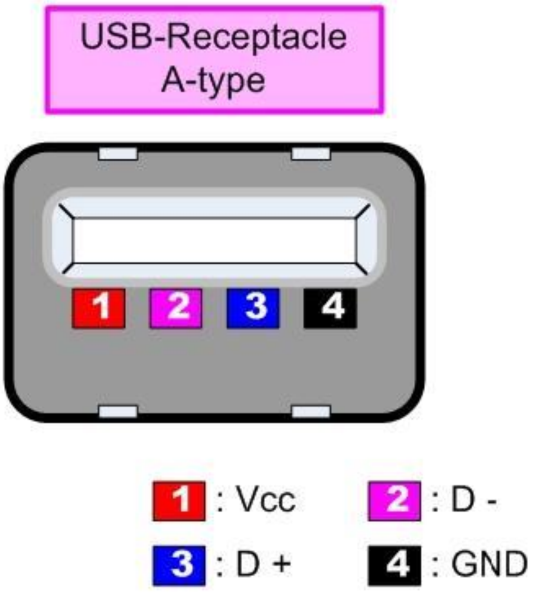
Pin	Signal	Description
1	TXD+	Physical transmit or receive signal (+ differential)
2	TXD-	Physical transmit or receive signal (- differential)
3	RXD+	Physical transmit or receive signal (+ differential)
6	RXD-	Physical transmit or receive signal (- differential)

<table>
<tr><th>LED</th><th>Description</th></tr>
<tr><td>Left Green</td><td>LAN Connection Speed
<table>
<tr><th>Speed</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>10Base-T</td><td>H</td><td>OFF</td></tr>
<tr><td>100Base-TX</td><td>L</td><td>ON</td></tr>
</table></td></tr>
<tr><td>Right Yellow</td><td>LAN Connection Status
<table>
<tr><th>Speed</th><th>Pin State</th><th>LED Definition</th></tr>
<tr><td>No Link</td><td>H</td><td>OFF</td></tr>
<tr><td>Link</td><td>L</td><td>ON</td></tr>
<tr><td>Activity</td><td>Toggle</td><td>Blinking</td></tr>
</table></td></tr>
</table>

2.7.3 J2 : USB Host

J2 is connected to USB HUB ControllerEddy-S4M in Eddy-S4M. Below is its PIN specification.





2.7.4 RESET switch

Pin	Function	Description	I/O
PC16	nRESET	Polling input signal continually from external reset key, implement as below with checking the constant time of "Low." Less than 5 seconds: Reboot 5 seconds or more: Factory Default function	IN

2.7.5 J4, 5 : Expansion Header

Provide most function of Eddy-S4M with pin connector.
You can confirm the function with direct conjunction to Eddy-S4M-DK.

J4

Pin	Signal	Pin	Signal
1	DTxD	2	DRxD
3	TxD0#	4	RxD0#
5	RTS0	6	CTS0
7	DTR0	8	DSR0
9	DCD0	10	RI0
11	TxD1#	12	RxD1#
13	RTS1	14	CTS1
15	3.3V	16	3.3V
17	P3_TX+	18	P3_TX-
19	P3_RX+	20	P3_RX-
21	P4_TX+	22	P4_TX-
23	P4_RX+	24	P4_RX-
25	DDM	26	DDP
27	DM2	28	DP2
29	DM3	30	DP3
31	GND	32	GND
33	DM4	34	DP4
35	SDDATA0	36	SDDATA1
37	SDDATA2	38	SDDATA3
39	SDCMD	40	SDCLK
41	SDCDN	42	SDWP
43	TWCK	44	TWD
45	RDY#	46	nRESET (IN)

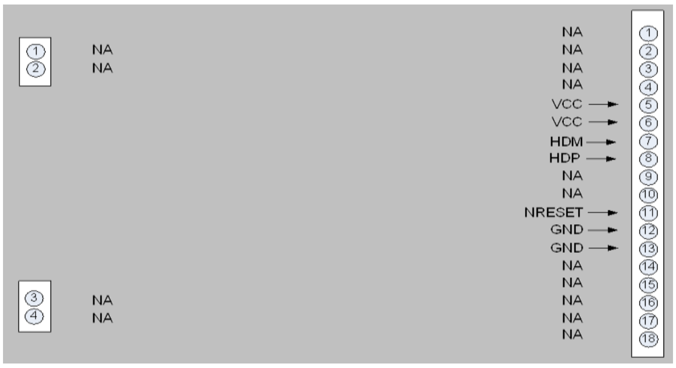
J5

Pin	Signal	Pin	Signal
1	LAN_TX+	2	LAN_RX+
3	LAN_TX -	4	LAN_RX-
5	LAN_LINK	6	LAN_Speed
7	PA5	8	PA22
9	PA30	10	NC
11	PB0	12	PB1
13	PB2	14	PB3
15	5V	16	5V
17	PB12	18	PB13
19	PB16	20	PB17
21	PB18	22	PB19
23	PB20	24	PB21
25	PB30	26	PB31
27	PC0	28	PC1
29	PC2	30	PC3
31	GND	32	GND
33	PC5	34	PC9
35	PC10	36	PC12
37	PC13	38	PC14
39	PC15	40	PC17
41	PC18	42	PC19
43	PC20	44	PC24
45	PC25	46	NRST (OUT)

2.8 Eddy-WiFi v3.0

(Note) For information about Eddy-WiFi v2.1 refer to the previous manual.

Eddy-WiFi is used with Eddy-CPU v2.5 series to add wireless LAN feature to use with security device, communication related device, modem, printer, industrial instruments, etc. Eddy-WiFi module supports IEEE 802.11b/g/n wireless standard.



Pin	Signal
1	NA
2	NA
3	NA
4	NA

Signal	Pin
NA	1
NA	2
NA	3
NA	4
VCC	5
VCC	6
HDM	7
HDP	8
NA	9
NA	10
NRESET	11
GND	12
GND	13
NA	14
NA	15
NA	16
NA	17
NA	18

LEFT	Description
1	NA
2	NA
3	NA
4	NA

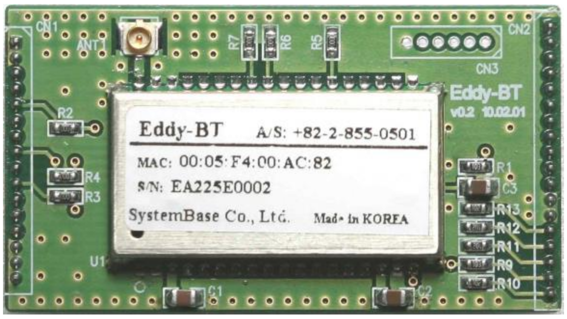
RIGHT	Description
1	NA
2	NA
3	NA
4	NA
5	VCC(3.3V)
6	VCC(3.3V)
7	USB Host Data(-)
8	USB Host Data(+)
9	NA
10	NA
11	H/W Reset
12	Ground
13	Ground
14	NA
15	NA
16	NA
17	NA
18	NA

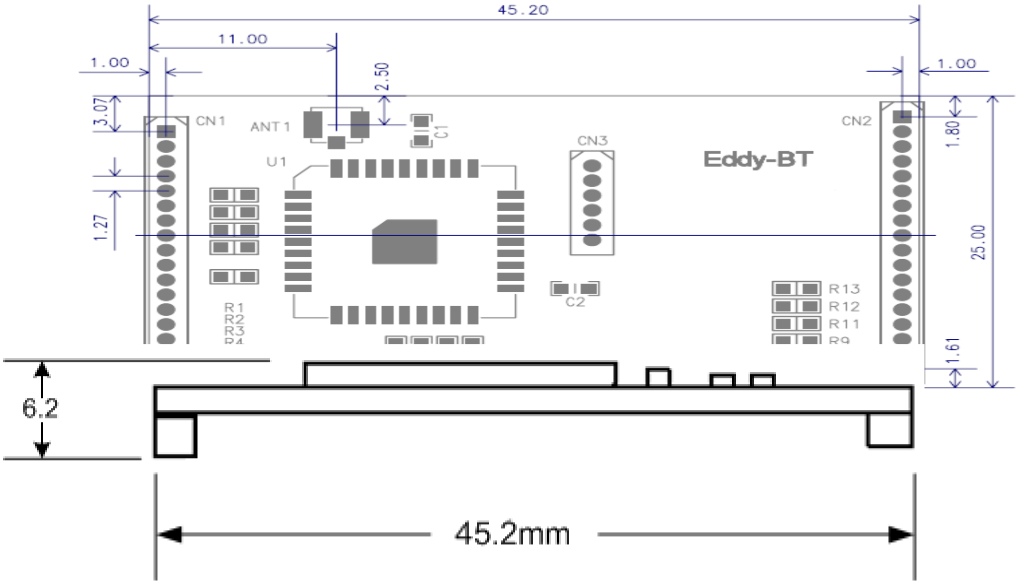
2.9 Eddy-BT v2.1

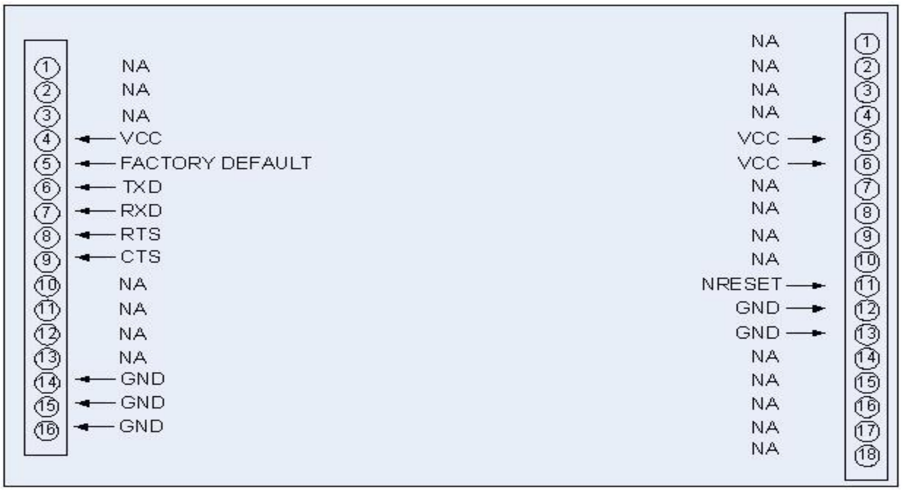
Eddy-BT module is based on Bluetooth 2.0 and supports communication distance of up to 100m. Linking to Eddy-CPU and Eddy-S4M, Eddy-BT module enables communication with various types of Bluetooth device in Bluetooth method. Eddy-BT module‘s communication interface supports serial method. To connect to Eddy-CPU, Eddy-S4M, it uses 4th serial port.

Since it is not considered to use Eddy-BT in Eddy‘s operating environment, it can lose data in case of using HW Flow Control. (4th port is composed to support RS422 or RS 485. Since it uses RTS/CTS signal line in Auto Toggle method, it cannot be used for HW flow control of RS232.)

For sample Eddy-BT source code, please refer to test_bluetooth.c.
To control Eddy-BT, refer to chapter 6 from Eddy_User_Guide.



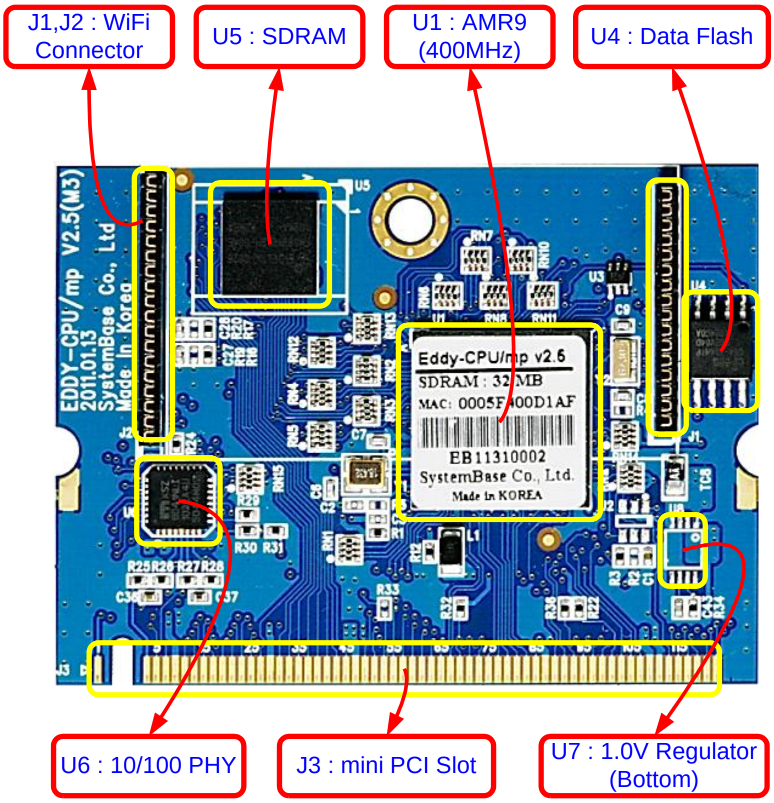




LEFT	Description
1	NA
2	NA
3	NA
4	VCC(3.3V)
5	Factory Reset
6	UART TXD
7	UART RXD
8	UART RTS
9	UART CTS
10	Pairing Signal
11	H/W Reset
12	NA
13	NA
14	Ground
15	Ground
16	Ground

RIGHT	Description
1	NA
2	NA
3	NA
4	NA
5	VCC(3.3V)
6	VCC(3.3V)
7	NA
8	NA
9	NA
10	NA
11	H/W Reset
12	Ground
13	Ground
14	NA
15	NA
16	NA
17	NA
18	NA

2.10 Eddy-CPU/mp v2.5



Eddy-CPU/mp V2.5 Mini PCI Card Type III System Connector Pinout(J3)

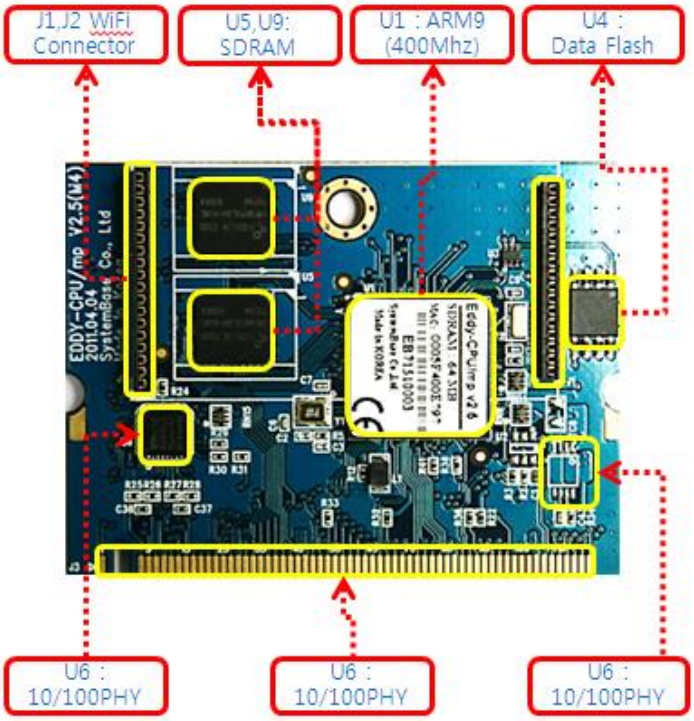
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	LAN_RX+	2	LAN_TX+	63	PB8	64	PB9
	Key		Key	65	PB10	66	PB11
3	LAN_RX-	4	LAN_TX-	67	PB12	68	PB13
5	LAN_Speed	6	LAN_LINK	69	DRXD	70	DTXD
7	FPG	8	RDY#	71	PB16	72	PB17
9	3.3V	10	GND	73	PB18	74	PB19
11	D0	12	D1	75	PB20	76	PB21
13	D2	14	D3	77	PB22	78	PB23
15	D4	16	D5	79	PB24	80	PB25
17	D6	18	D7	81	PB26	82	PB27
19	D8	20	D9	83	PB28	84	PB29
21	D10	22	D11	85	PB30	86	PB31
23	D12	24	D13	87	3.3V	88	GND
25	D14	26	D15	89	PC0	90	PC1
27	NRD	28	NWE	91	PC2	92	PC3
29	3.3V	30	GND	93	PC5	94	PC8
31	A0	32	A1	95	PC9	96	PC10
33	A2	34	A3	97	PC12	98	PC13
35	A4	36	A5	99	PC14	100	PC15
37	A6	38	A7	101	nRESET	102	PC17
39	A8	40	A9	103	PC18	104	PC19
41	A10	42	A11	105	PC20	106	PC21
43	A12	44	A13	107	PC22	108	PC23
45	A14	46	A15	109	3.3V	110	GND
47	3.3V	48	GND	111	GND	112	PC26

49	PA4	50	PA22	113	TWCK	114	TWD
51	PA5	52	PA30	115	DDP	116	DDM
53	PA31	54	NRST	117	HDPA	118	HDPB
55	PB0	56	PB1	119	HDMA	120	HDMB
57	PB2	58	PB3	121	NAND_OE	122	A21
59	PB4	60	PB5	123	NAND_WE	124	A22
61	PB6	62	PB7				

J2	
Pin	Signal Name
1	PB0
2	PB1
3	PB2
4	PB3
5	3.3V
6	3.3V
7	BHDM, USB Host Data(-)
8	BHDP, USB Host Data(+)
9	PA31 / TXD4
10	PA30 / RXD4
11	NRST
12	GND
13	GND
14	PA9 / WPID0
15	PC6 / WPID1
16	PC7 / WPID2
17	NC
18	NC

J1	
Pin	Signal Name
1	NC
2	NC
3	3.3V
4	3.3V
5	PC25 / BT_Factory
6	PB10 / TXD3
7	PB11 / RXD3
8	PC8 / RTS3
9	PC10 / CTS3
10	PC24 / BT_MODE
11	NRST
12	GND
13	GND
14	NC
15	NC
16	NC

2.11 Eddy-CPU/mp v2.5 32bit



Eddy-CPU/mp v2.5 32bit Mini PCI Card Type III System Connector pinout (J3)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	LAN_RX+	2	LAN_TX+	63	PB8	64	PB9
	Key		Key	65	PB10	66	PB11
3	LAN_RX-	4	LAN_TX-	67	PB12	68	PB13
5	LAN_Speed	6	LAN_LINK	69	DRXD	70	DTXD
7	FPG	8	RDY#	71	PB16	72	PB17
9	3.3V	10	GND	73	PB18	74	PB19
11	D0	12	D1	75	PB20	76	PB21
13	D2	14	D3	77	PB22	78	PB23
15	D4	16	D5	79	PB24	80	PB25
17	D6	18	D7	81	PB26	82	PB27
19	D8	20	D9	83	PB28	84	PB29
21	D10	22	D11	85	PB30	86	PB31
23	D12	24	D13	87	3.3V	88	GND
25	D14	26	D15	89	PC0	90	PC1
27	NRD	28	NWE	91	PC2	92	PC3
29	3.3V	30	GND	93	PC5	94	PC8
31	A0	32	A1	95	PC9	96	PC10
33	A2	34	A3	97	PC12	98	PC13
35	A4	36	A5	99	PC14	100	PC15
37	A6	38	A7	101	nRESET	102	PC17
39	A8	40	A9	103	PC18	104	PC19
41	A10	42	A11	105	PC20	106	PC21
43	A12	44	A13	107	PC22	108	PC23
45	A14	46	A15	109	3.3V	110	GND
47	3.3V	48	GND	111	GND	112	PC26
49	PA4	50	PA22	113	TWCK	114	TWD
51	PA5	52	PA30	115	DDP	116	DDM

53	PA31	54	NRST	117	HDPA	118	HDPB
55	PB0	56	PB1	119	HDMA	120	HDMB
57	PB2	58	PB3	121	NAND_OE	122	A21
59	PB4	60	PB5	123	NAND_WE	124	A22
61	PB6	62	PB7				

J2	
Pin	Signal Name
1	PB0
2	PB1
3	PB2
4	PB3
5	3.3V
6	3.3V
7	BHDM, USB Host Data(-)
8	BHDP, USB Host Data(+)
9	PA31 / TXD4
10	PA30 / RXD4
11	NRST
12	GND
13	GND
14	PA9 / WPID0
15	PC6 / WPID1
16	PC7 / WPID2
17	NC
18	NC

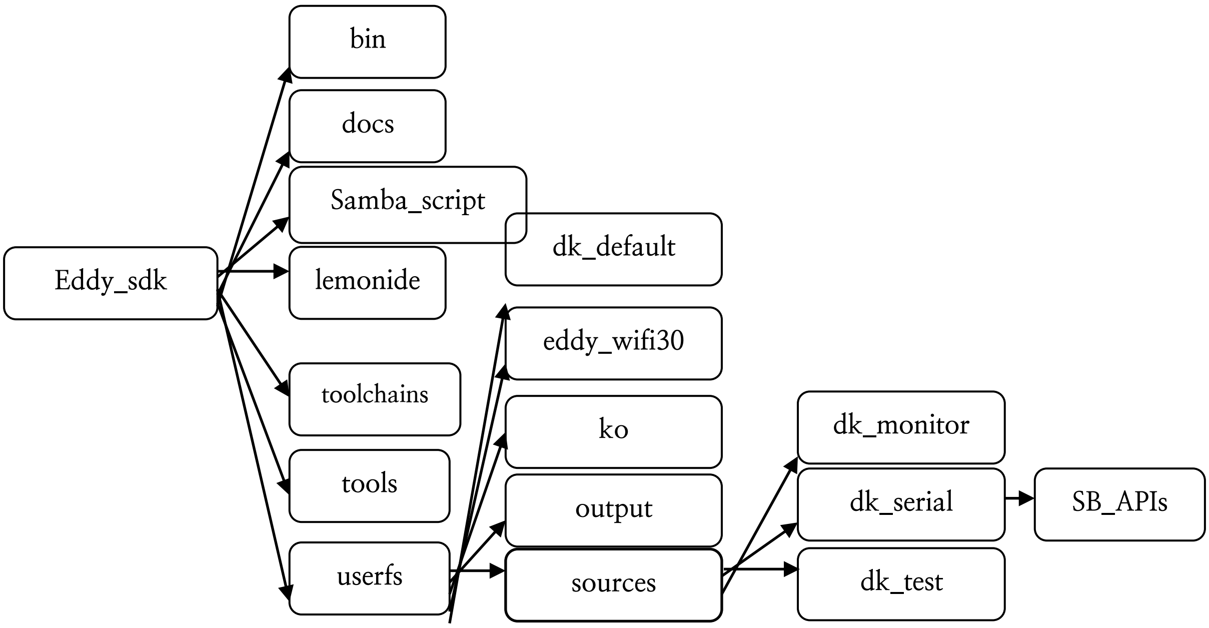
J1	
Pin	Signal Name
1	GND
2	GND
3	GND
4	3.3V
5	PC25 / BT_Factory
6	PB10 / TXD3
7	PB11 / RXD3
8	PC8 / RTS3
9	PC10 / CTS3
10	PC24 / BT_MODE
11	NRST
12	3.3V
13	3.3V
14	GND
15	GND
16	GND

Chapter 3. Development Environment

This chapter explains the process of application programming and other important notes.
The folder structures for SDK are as follows.

> Note:
>
> All material related to Eddy including documents, reference sources and utilities are periodically updated at www.embeddedmodule.com without prior notice. Please visit and download the latest updates from the site.

3.1 Structure of the folders containing source codes



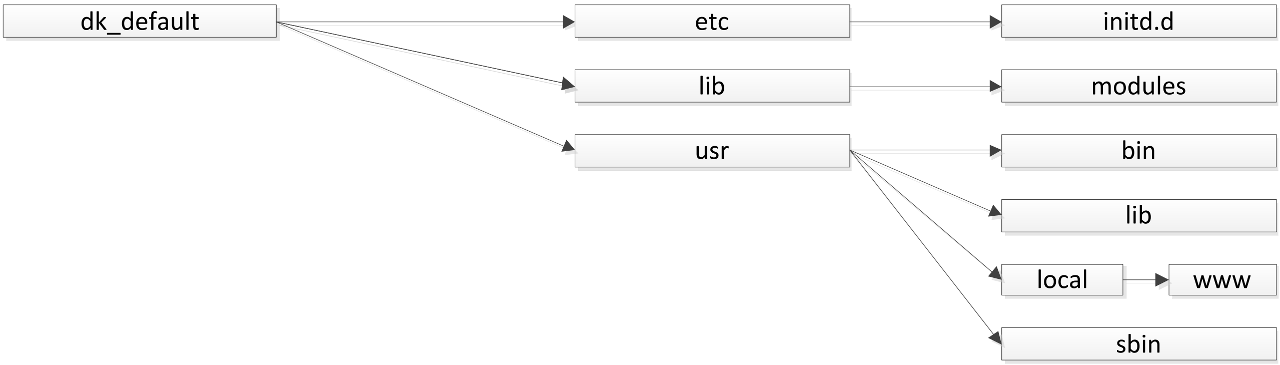
bin
Repository for Cygwin binary and library to use cross toolchain

toolchains
Repository of cross toolchain for Eddy software development

tools
Repository for windows tools used for Eddy software development

lemonide
Eddy software development based on Eclipse Integrated Development Environment (IDE)

userfs/dk_default



Folders are divided by etc, usr, lib according to basic use of file system in Eddy DK. In this file system, when you create a file and build a firmware in this location, it will be compressed as Eddy_sdk/userfs/output/dk_default.zip. If you upload this to your Eddy, it will be applied to Eddy file system.

- etc/initd.d

This is where the scripts for initial execution when Eddy boots. You can create scripts that you would like to run the application or modules.

- lib/modules

Device driver modules used in Eddy DK are located.

- usr/bin

Basically includes applications for Eddy DK, but user applications can be added also.

- usr/lib

Basically includes library for Eddy DK, but user library can be added also.

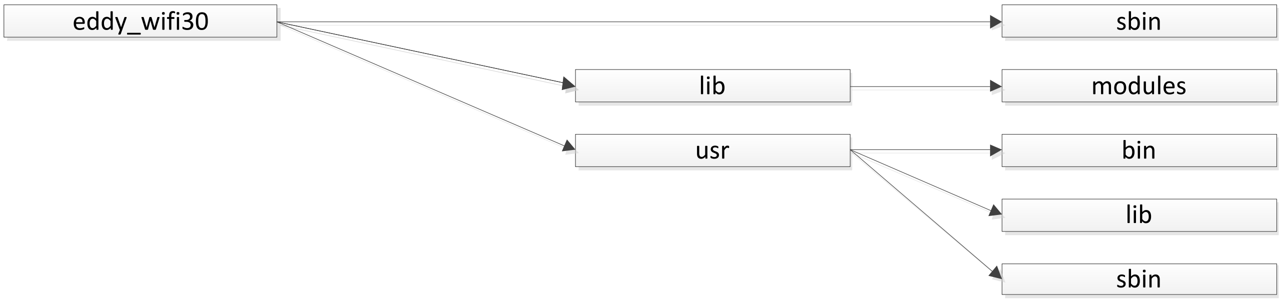
- usr/local/www

Web page and configuration files for web pages are stored which is displayed when you connect Eddy DK with a web browser. To get more information to customize these settings, refer to ‘chapter 8 Modify the Web Page’.

- usr /sbin

System programs used in Eddy DK are located here.

eddy_wifi30



- sbin
  The iwconfig needed for wireless connection is located.

- lib/modules
  Drivers module used for Eddy_wifi30 are located.

- usr/bin
  Commands for connecting wifi are located. You can add customized functions and include them.

- usr/lib
  Basic libraries for Eddy_wifi30 are included. You can add your own libraries to include them.

- usr/sbin
  System commands used in Eddy_wifi30 are located.

3.2 Language in use

Eddy-DK application should be composed with C language. All example source codes provided is composed in C language. You can use more than one source file if you are using C programming Language. If you are familiar with programming with ANSI C, there will be no difficulties creating applications for Eddy.

3.3 Development Environment

To develop with Eddy DK, you need Linux or Windows operating system.
Following table shows tested the compatible environment under Linux and Windows.

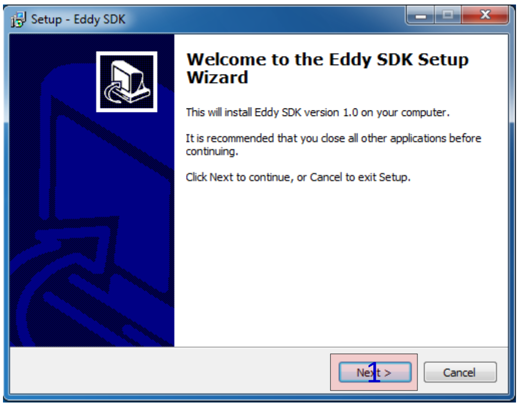
Windows	Linux
Windows 7 32bit Windows XP SP3 32bit Windows 2000 Windows 2003	Red Hat 9.0 Fedora Core 4, 5, 6 SUSE Enterprise Server 10.2 Ubuntu 6.x, 7.x Debian 4.0 CentOS 4.5 Asianux 3rd edition

3.4 Installing in Windows

This chapter will describe how to install Eddy Development Environment on Windows host. The explanation of this manual based on Windows XP. To establish Eddy‘s integrated development environment, LemonIDE, please refer to “LemonIDE_User_Guide” for further instructions.

3.4.1 Installing Eddy SDK

Run Eddy SDK.exe, and install it by following order.



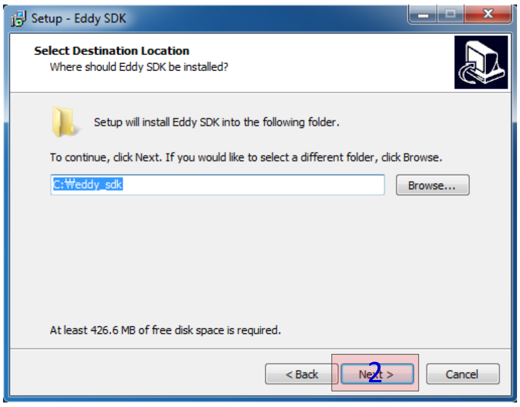
Setup - Eddy SDK

Welcome to the Eddy SDK Setup Wizard

This will install Eddy SDK version 1.0 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.



Setup - Eddy SDK

Select Destination Location

Where should Eddy SDK be installed?

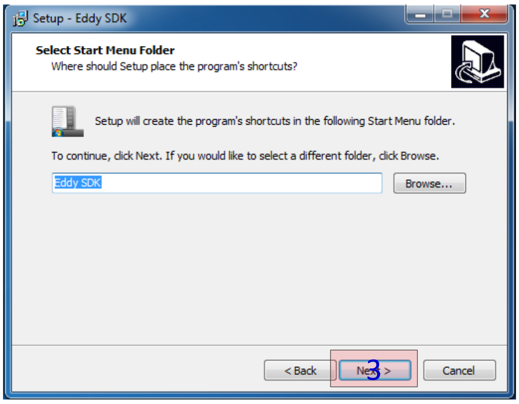
Setup will install Eddy SDK into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:₩eddy_sdk

At least 426.6 MB of free disk space is required.

When you run setup file, you will see an installation window as shown above. Click Next in 1. Next screenshot shows where SDK will be installed the default folder is C:\eddy_s. Click Next in 2.



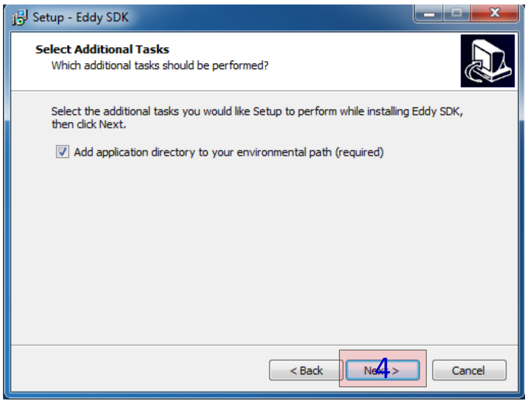
Setup - Eddy SDK

Select Start Menu Folder
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Eddy SDK



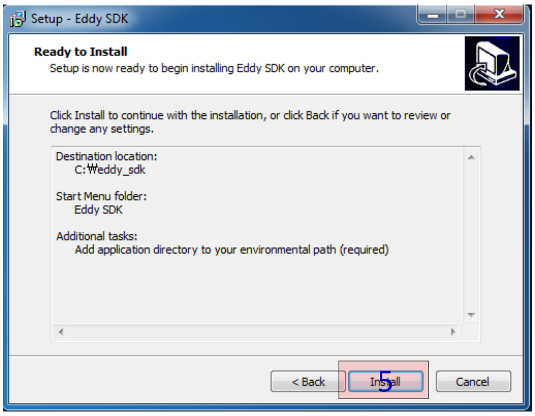
Setup - Eddy SDK

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Eddy SDK, then click Next.

☑ Add application directory to your environmental path (required)

Choose the folder name for start menu in Windows and click Next in 3. Default name is set to Eddy_SDK. Next one shows that for environmental path, setup will add application directory to the current path to access the files in the directory. Click Next in 4.



Setup - Eddy SDK

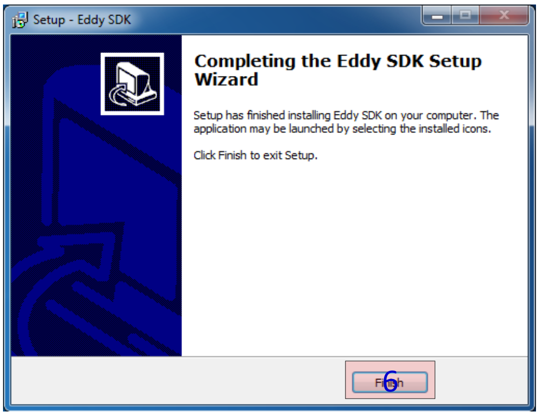
Ready to Install
Setup is now ready to begin installing Eddy SDK on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:₩eddy_sdk

Start Menu folder:
Eddy SDK

Additional tasks:
Add application directory to your environmental path (required)



Setup - Eddy SDK

Completing the Eddy SDK Setup Wizard

Setup has finished installing Eddy SDK on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

Depends on your system, it takes few minutes to complete installing Eddy SDK. When it is complete, click Finish in 6.

3.5 Removing Development Environment

Development Environment can be removed by simply deleting the folder where installed files are located.

3.5.1 Removing Windows Development Environment

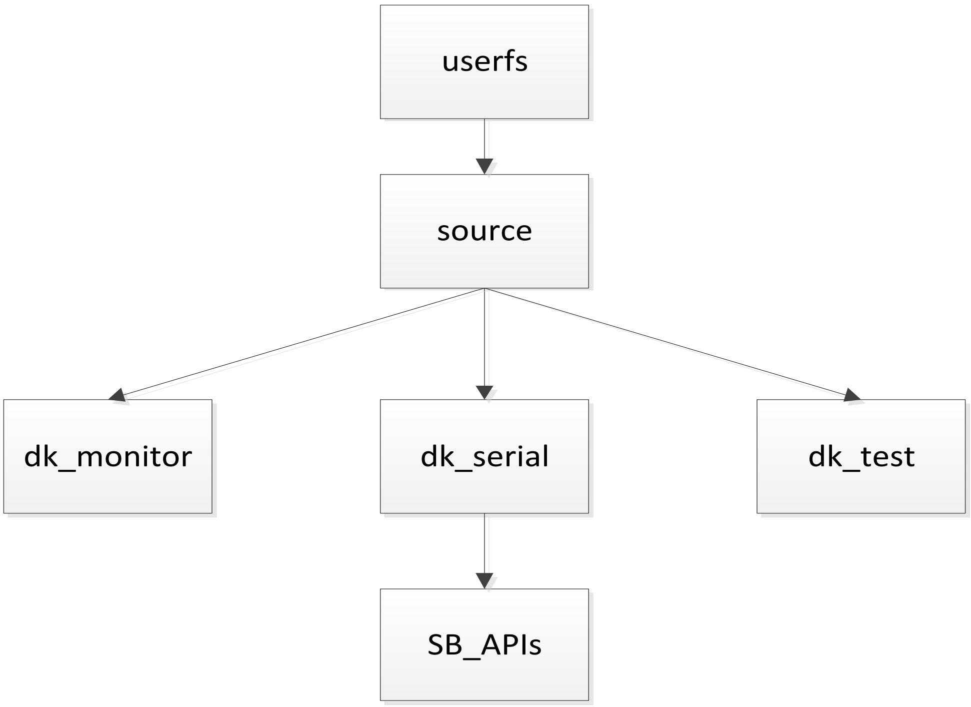
Uninstall Eddy SDK in Windows to remove Eddy development environment.

Chapter 4. Compiling an Application Program

4.1 Introduction to writing a program

Write an application program, compile it and upload it into Eddy to see whether it is running without any problem. It shows how to save the firmware image in Eddy flash memory.

When writing a new application program, refer to the sample source codes in Eddy_sdk/userfs/source folder. There are no dedicated application only source code for Device Server, but developers can refer to the sample codes to write a customized application.



dk_monitor
There is a sample that shows how to display value in LCD for Eddy DK. Depending on a keypad input, change in result can be seen from LCD.

dk_serial
Sample for Ethernet and RS232, RS422/485 type serial communications are included. Subfolders in /SB_APIs includes API samples for developers who wants to implement with themor just to check those functions.

dk_test
Samples source codes to test each devices in Eddy DK are located. Developers can refer to these samples to write and test their codes.

4.2 How to write an application program

Create application program that can be run from Eddy. First, write a hello.c as below under /userfs/source/dk_test/. (This example is based on de_test folder, if necessary, include dk_monitor or dk_serial folder)

```
#include <stdio.h>
int main()
{
        int i;
        sleep(20);
        for(i=0;i < 5; i++)
        {
                printf("Hello World!!₩n");
                sleep(1);
        }
        return 0;
}
```

4.3 How to make a Makefile

To compile an application program, related information should be registered under /dk_test/Makefile/. (Same with other folders.) Following file is from the Makefile under /userfs/source/dk_test/.
Marked in red is what are added to the environment setting for application compile.

```
CC=arm-linux-gcc

all: dk_test hello

clean:
                @rm -f *.o
                @rm -f dk_test

install: dk_test
                @cp dk_test ../../dk_default/usr/bin

dk_test: dk_test.o
```

When using /dk_serial/Makefile, use as shown below instead of using ‘all: dk_test hello’

```
                              …
TARGETS = dk_serial serialconf dk_serial_test hello
all : $(TARGETS)
        @echo done
                              …
```

4.4 Compile an applicationprogram

Compile the application program to execute on Eddy after registering the compile environment to the "Makefile".

4.4.1 Compile in Windows

Enter "make" command through CMD (command prompt) on the folder where "Makefile" is located. As shown below, if a compile is successfully completed, execution file named "Hello_World" would be created. Of course, as this file was cross-compiled, it cannot run on Windows environment. Upload this file to Eddy using FTP to execute the file on Eddy, (Files uploaded with FTP will not permanently saved in Eddy.).
This will be further explained on the next chapter (Chapter 5 User File System).

```
C:₩eddy_sdk₩userfs₩sources₩dk_test>make hello
arm-linux-gcc      -c -o hello.o hello.c
arm-linux-gcc    hello.o    -o hello

C:₩eddy_sdk₩userfs₩sources₩dk_test>ls
Makefile    dk_test.c    eddy_gpio.h    hello.c
Makefile~    eddy_adc.h    hello          hello.o
```

4.4.2

4.5 Run in Eddy

To run an application on Eddy, there are several methods. First method is to convert an application as a firmware and loads it into the flash memory area and execute. However, this method is not recommended for developing phase of application, since it is time consuming task. Second method is to load and execution file of an application to RAM type file system by using the FTP Server on Eddy DK, and execute it from there. This method is suitable for developing phase of application; however the application loaded to Eddy will be deleted when the power is disconnected.
The LemonIDE integrated developing environment provides advanced solution. LemonIDE debugging tool supports the direct transmission of compiled applications to Eddy. By using this tool, the user can execute and check the result instantly on site.
If you wish to use LemonIDE, please refer to "LemonIDE_User_Guide".

4.5.1 Run after uploading into Eddy

Connect to Eddy by using FTP. ID and password for FTP server are same as the one using with telnet connection.
The example below shows how to upload an example file, 'hello', to /tmp folder of Eddy on Linux using FTP.
When uploading a file, "bin" command must be entered first for binary mode. For uploading enter "put <file name> on the command line. Following is an example how you can connect Eddy through FTP from Linux.

```
[root@localhost dk_test]# ftp 192.168.0.223
Connected to 192.168.0.223 (192.168.0.223).
220 Welcome to Eddy FTP service.
Name (192.168.0.223:root): eddy
331 Please specify the password.
Password:********
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put hello
local: hello remote: hello
227 Entering Passive Mode (192,168,0,223,234,124).
150 Ok to send data.
226 Transfer complete.
5014 bytes sent in 0.000448 secs (11191.96 Kbytes/sec)
ftp> bye
```

FTP program from the Command Prompt. Following is an example related to it.

```
C:₩eddy_sdk₩userfs₩sources₩dk_test>ftp 192.168.0.223
Connected to 192.168.0.223.
220 Welcome to Eddy FTP service.
User (192.168.0.223:(none)): eddy
331 Please specify the password.
Password: ********
230 Login successful.
ftp> bin
200 Switching to Binary mode.
ftp> put hello
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp: 5006 bytes sent in 0.00Seconds 5006000.00Kbytes/sec.
```

When the transmission is completed, a user can check the file using Telnet terminal connected Eddy.
The file is executable using “chmod” command; however the mode has to be switched to executable.

After switching to Executable Mode, execute the file by entering "/hello_world".
To terminate a program, press "Ctrl" and "C" keys simultaneously.

4.5.2 Set the Application to run automatically when Eddy boots

```
# cd /tmp
# ls
eddy.cfg      ifstate      log          login.pw      resolv.conf   utmp
hello         klogd.pid    login.id     messages      syslog.pid    vsftpd.log
# chmod 777 hello
# ./hello
hello world!!
hello world!!
hello world!!
hello world!!
```

If the application is successfully executed on Eddy, make a firmware image and load to Flash memory of Eddy to execute on booting.

In chapter 4.4 Compile an application program, it described that when you run make install from the application program it will create a user created executable file under folder for firmware file. To successfully do this task, refer to 4.3 Makefile.

To do this task manually, copy executable file to /eddy_sdk/userfs/dk_default/usr/bin/ for Eddy DK, and for Eddy-WiFi copy it to /eddy_sdk/userfs/eddy_wifi30/usr/bin/.

If you build a file system from this status, user application program will include in file system. If you would like to know how to apply it to Eddy, refer to Chapter 5. Here we will go over with how to run an application program from Eddy automatically.

There are serveral ways to automatically run the user application program. We will go over with two easier way to do it.

1. First, you can use the initialization script in Eddy.

   There are scripts that will automatically start in dk_default/etc/initd.d folder. If you add a command that executes user application, it will run automatically. If you do not need any files to be run automatically, you can skip this.

```
#!/bin/sh
/usr/bin/hello &
```

   The reason that & is added at the end of the command is, to prevent Eddy for waiting infinitely when hello is executed. If you do not add &, Eddy will not continue initialization and try to standby endlessly.

   There are rules for naming files in this folder. Following shows files under /dk_default/etc/initd.d/ for Windows. U11apps2 is the exemplary script file containing the content related to above and it is written in UXXFilename form.

```
C:\eddy_sdk\userfs\dk_default\etc\init.d>ls
U01i2c   U02gpioinit   U03adcinit   U04lcdinit   U05ipv6   U06usb   U10apps   U11apps2

C:\eddy_sdk\userfs\dk_default\etc\init.d>
```

The first letter U is used for user created script. Next two digits XX are numbers that indicate the order of the execution. The last, Filename, is just as it states, the name of the script file. In order for the script file to run successfully, first three UXX must be named in such way.

If you complete all the tasks up to this point, you will be able to automatically run Eddy when it is booted. In chapter5 User File System, you will learn how to apply modified file system in Eddy.

2. Second, you can use the user-modified initialization script

During booting, Eddy runs the initialization script defined in environmental variable "userinit".

This script executes before anything else, so it is good place to include high priority files to be executed. Those files are executed before the file system, therefore, if you try to use the commands in the file system, you will see the following message.
The dk_default.zip was executed before unzip is applied and showing "not found".

```
Starting logging: OK
Run /flash/userinit
/flash/userinit: line 2: /usr/bin/hello: not found
Unzip /flash/eddy_wifi30.zip at /
Unzip /flash/dk_default.zip at /
Starting network...
```

As shown here, if you are trying to use userinit, you have to upload an executable file in /flash folder to run it.

userinit=/flash/userinit

When the environmental variable is set as above, /flash/userinit will be used for initialization scrip and while booting, user-defined initialization script will be called.

While pressing and holding the reset button when you supply power to Eddy, it will enter the bootloader configuration menu. (Hold for more than 5 seconds.) You can use printenv command to check the values in userinit.

```
Eddy> printenv userinit
userinit=/flash/userinit
Eddy>
```

To remove or add values to environmental variable setting in userinit under bootloader configuration, refer to the example below.

```
Eddy> setenv userinit
Eddy> saveenv
Saving Environment to dataflash...

Eddy> printenv userinit
## Error: "userinit" not defined
Eddy> setenv userinit /flash/userinit
Eddy> saveenv
Saving Environment to dataflash...

Eddy> printenv userinit
userinit=/flash/userinit
Eddy>
```

Userinit script files set in userinit should be in /flash folder and this allows user to write a script or upload directly from Eddy.

Login to Eddy and move to /flash folder. Use the vi editor to create userinit file shown as below.

```
# cd /flash/
# vi userinit
```

Use vi editor to write a content as shown below.

```
#!/bin/sh
/flash/hello &
```

Change file permission to 777 as shown below.

```
# chmod 777 userinit
#
```

After rebooting, check whether the program works as intended with the command shown below.

```
# ps
  PID USER       VSZ STAT COMMAND
    1 root      1168 S    init
    2 root         0 SW<  [kthreadd]
  …
  298 root       588 S    /flash/hello
  …
  611 root      1148 S    usleep 500000
  612 root      1148 S    usleep 500000
  613 root      1156 R     ps
# Hello World!!
Hello World!!
Hello World!!
Hello World!!
Hello World!!

#
```

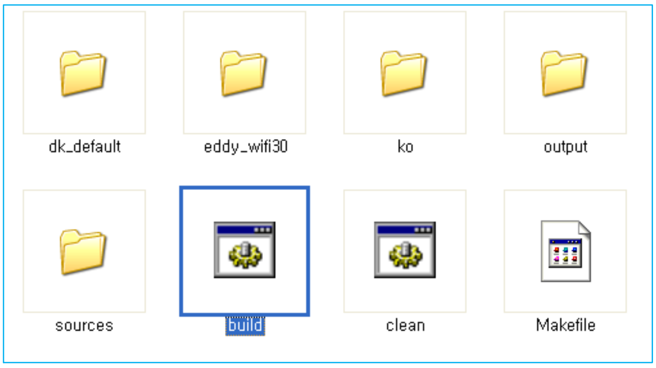
Chapter 5. Create User File System

On the previous chapter, we explained how to make and compile application program with sample program. This chapter introduces methods to create a firmware which permanently saves the application into the Eddy module and apply it to hardware of Eddy.

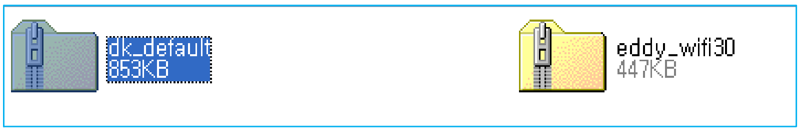
5.1 Compressing file system

When you run Makefile under eddy_sdk/userfs, Eddy file system under dk_default and eddy_wifi30 folders will be compressed to ZIP format in / eddy_sdk/userfs/output/.
In Eddy, the build and the clean execution files will be provided to run Makefile easily for GUI under Windows. Following is an execution example for GUI under Windows.

When you run build, file system will be compressed under output folder. When you run clean, compressed file created by build will be deleted.



When build is executed successfully, you will see created files as below in the output folder.



In Linux, you can execute it with make command. The following is the example for it.

```
[root@localhost userfs]# make
  adding: etc/ (stored 0%)
  adding: etc/init.d/ (stored 0%)
                    …
  adding: usr/bin/wifiup (deflated 54%)
  adding: usr/bin/wpa_config (deflated 53%)
[root@localhost userfs]# cd output/
[root@localhost output]# ls
dk_default.zip   eddy_wifi30.zip
```

The following shows how you can execute in command prompt in Windows.

```
C:\eddy_sdk\userfs>make
updating: etc/ (stored 0%)
updating: etc/init.d/ (stored 0%)
                                   …
updating: usr/sbin/wpa_supplicant (deflated 51%)

C:\eddy_sdk\userfs>cd output
C:\eddy_sdk\userfs\output>ls
dk_default.zip    eddy_wifi30.zip
```

5.1 Apply file system in Eddy

5.1.1 How to upload a file using FTP service

Let's user FTP to apply compressed file system from above in Eddy. The following example shows how you can use FTP to upload the compressed file system into Eddy.

```
C:\eddy_sdk\userfs\output>ls
dk_default.zip    eddy_wifi30.zip

C:\eddy_sdk\userfs\output>ftp 192.168.0.223
Connected to 192.168.0.223.
220 Welcome to Eddy FTP service.
User (192.168.0.223:(none)): eddy
331 Please specify the password.
Password:********
230 Login successful.
ftp> bi
200 Switching to Binary mode.
ftp> put dk_default.zip
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp: 875135 bytes sent in 4.09Seconds 213.76Kbytes/sec.
ftp> bye

C:\eddy_sdk\userfs\output>
```

Use telnet to connect to Eddy. Following example shows how you can move uploaded file system to /flash in Eddy.

```
# pwd
/tmp
# ls
dk_default.zip    klogd.pid         login.pw          syslog.pid
eddy.cfg            log                 messages          utmp
ifstate           login.id          resolv.conf       vsftpd.log
# mv dk_default.zip /flash/
# cd /flash/
# ls
dk_default.zip
eddy_wifi30.zip
upgrade.log
#
```

Configure bootloader to apply the file system

With Eddy bootloader, developers can apply modified file system easily. When you press and hold reset button while supplying power to Eddy, it will enter bootloader configuration menu. (Hold the button for more than 5 seconds.)

Bootloader is only displayed from system standard output, the console port. Connect debug port from Eddy DK board to serial port in your PC. Use HyperTerminal or equivalent serial emulator and set the speed to 115k, none, 8, 1 to check the result from it.

The following will be displayed when you successfully enter the bootloader.

```
RomBOOT


U-Boot 1.3.4-svn12 (Nov 15 2012 - 18:41:00)

DRAM:   32 MB
In:     serial
Out:    serial
Err:    serial
Net:    macb0
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Eddy>
```

To apply uploaded file system, /flash/dk_default.zip, through FTP, to the root file system, configure settings as following.

```
Eddy> setenv /flash/dk_default.zip /
Eddy> saveenv
Saving Environment to dataflash...

Eddy> printenv
baudrate=115200
ethaddr=00:05:f4:00:90:57
ethact=macb0
wan=static
…
stdout=serial
stderr=serial
/flash/dk_default.zip=/

Environment size: 938/65532 bytes
Eddy>
```

The setenv command is used to set the environmental variable the format to use it is setenv [environmental variable]

[variable].
As shown above, when you use setenv [filesystem.zip] [path to uncompressed] to execute this command, dk_default.zip will be applied to the top most /. When you change the environmental variables, use saveenv command to save the changed value.

To check the modified value, use printenv.
Press the reset button to run the applications automatically you created previously. The results will only be displayed from the console port. The console port in Eddy is debug port for Eddy DK therefore, you cannot use telnet to check the result of your application program. Connect debug port in Eddy DK to the serial port in your PC and use HyperTerminal or equivalent serial emulator program and set the communication speed to 115k, none, 8, 1 to check the result for your application program.

```
# hello world!!
hello world!!
hello world!!
hello world!!
hello world!!
```

To cancel applied file system, enter bootloader mode and use printenv command to check the current settings.

```
RomBOOT


U-Boot 1.3.4-svn12 (Nov 15 2012 - 18:41:00)

DRAM:   32 MB
In:     serial
Out:    serial
Err:    serial
Net:    macb0
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Eddy>printenv
…
userinit=/flash/userinit
/flash/dk_default.zip=/
stdin=serial
stdout=serial
stderr=serial
Environment size: 938/65532 bytes
```

You can see that variable settings are set to /flash/dk_default.zip as shown above. Following is an example how you can check and remove the environmental variable.

```
Eddy> setenv /flash/dk_default.zip
Eddy> saveenv
Saving Environment to dataflash...

Eddy> printenv
baudrate=115200
ethaddr=00:05:f4:00:90:57
ethact=macb0
```

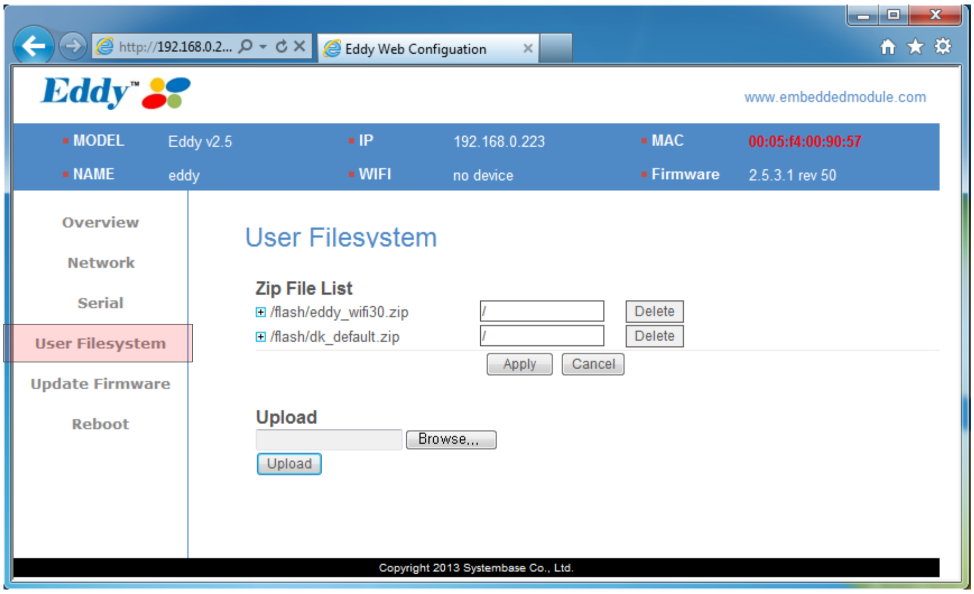
```
…
userinit=/flash/userinit
stdin=serial
stdout=serial
stderr=serial

Environment size: 914/65532 bytes
Eddy>
```

Since the variable is removed, applied file system is not working anymore.

5.1.2 How to use Eddy with a Web Browser

Eddy provides a service where you can use any web browser to configure its settings. When you connect to the IP assigned to Eddy, it will show as following. When you select User Filesystem, current files in /flash folder will be displayed under Zip File List.



MODEL	Eddy v2.5	IP	192.168.0.223	MAC	00:05:f4:00:90:57
NAME	eddy	WIFI	no device	Firmware	2.5.3.1 rev 50

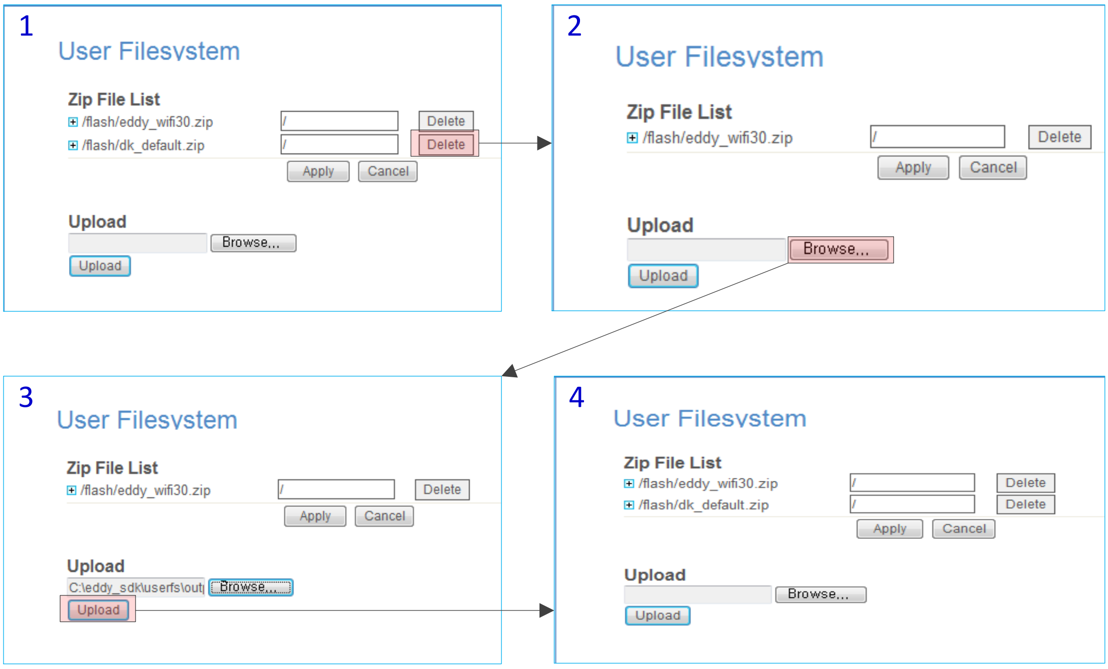
User Filesystem

Zip File List

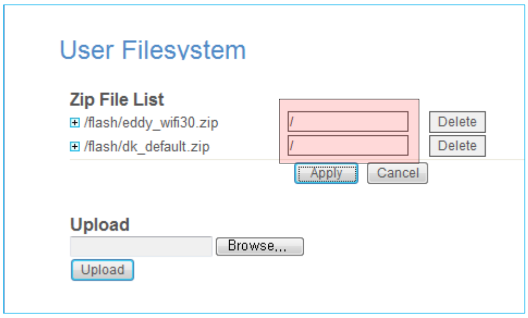
- /flash/eddy_wifi30.zip /
- /flash/dk_default.zip /

Upload

Copyright 2013 Systembase Co., Ltd.



In the first page above, you can remove files from Zip File List simply pressing delete button. The second page shows that it is removed from the Zip File List when delete button is pressed. The bottom of the second page provides a feature to upload current file in PC to /flash in Eddy. Click on Browse button to choose a file to upload. You will see a file path as in third page. Click on Upload button to see that your file has been uploaded and displayed in the Zip File List.

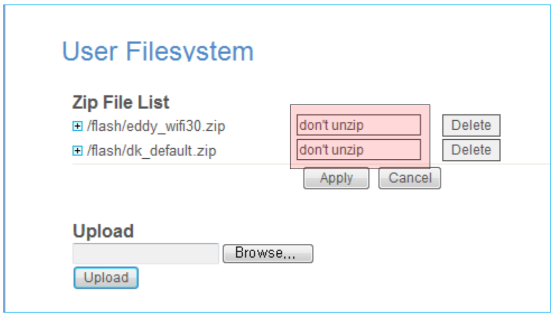


User Filesystem

Zip File List

⊞ /flash/eddy_wifi30.zip / Delete

⊞ /flash/dk_default.zip / Delete



User Filesystem

Zip File List

⊞ /flash/eddy_wifi30.zip don't unzip Delete

⊞ /flash/dk_default.zip don't unzip Delete

Even if the file is in the Zip File List, it is not always uncompressed. eddy_wifi30.zip and dk_default.zip files are in the file system which is by default set to uncompressed in /. When you delete this value and click on Apply button, it will show “don’t unzip” and file uncompressing will not be applied. You do not have to remove the file when you uploaded a user file system that you do not want to apply it immediately. You can simply use this avoid using it for now.

5.3 Firmware upgrade

Upload a firmware file into Eddy and save it in the flash memory. There are 4 ways to upgrade the firmware.

By FTP — Use FTP to connect to Eddy and upload a firmware image. Then, use telnet to save it in the flash memory with upgrade command.

By Web browser — Connect through Web server in Eddy and user Upgrade tab to save a firmware image in the flash memory. For more information, refer to Eddy-User Guide.

By bootloader — When rebooting enter bootloader mode and save a firmware image in the flash memory using debug port in Eddy DK board. For more information, refer to chapter 9, section 1 System recovery via bootloader.

By USB — Use the USB client port in Eddy DK board to save a firmware image in the flash memory. This is only available from Windows host. For more information, refer to chapter 9, section 2 System recovery via port.

This section show how you can use FTP to update.

In Windows, you can use CMD (Command Prompt) and pre-installed FTP program from Windows to upload. A complete image, eddy-fs-2.x.x.x.bin , can be uploaded to /tmp/ in Eddy via FTP.

```
C:\eddy_sdk\samba_script>ftp 192.168.0.212
Connected to 192.168.0.212.
220 Welcome to Eddy FTP service.
User (192.168.0.212:(none)): eddy
331 Please specify the password.
Password:
230 Login successful.
ftp> bi
200 Switching to Binary mode.
ftp> put eddy-2.5.3.1.bin
```

```
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
ftp: 8388608 bytes sent in 7.09Seconds 1182.49Kbytes/sec.
ftp> bye

C:\eddy_sdk\samba_script>
```

Use telnet to connect to Eddy and check ‘eddy-2.x.x.x.bin’ file at /tmp/ folder.
Upgrade the firmware with ‘upgrade eddy-2.x.x.x.bin’ command.

```
# pwd
/tmp
# ls
eddy-2.5.3.1.bin   klogd.pid      login.pw      syslog.pid
eddy.cfg           log            messages      utmp
ifstate            login.id       resolv.conf   vsftpd.log
# upgrade eddy-2.5.3.1.bin
Firmware version is 2.5.2.0 rev 34
Erasing 64 Kibyte @ 1f0000 - 100% complete.
Writing firmware...done
Verifying...ok
read done
Updating environments is done.
success
# exit
```

Chapter 6. Bootloader

This chapter covers about environmental settings and commands regarding the bootloader. To enter bootloader mode, supply power and press the reset button for 3 seconds. You should press the button right after you power on the unit, but if it is difficult, you can press the reset button before you supply the power, but hold the button for 3 seconds after the power is supplied.

```
RomBOOT



U-Boot 1.3.4-svn12 (Nov 15 2012 - 18:41:00)

DRAM:  32 MB
In:    serial
Out:   serial
Err:   serial
Net:   macb0
macb0: Starting autonegotiation...
```

6.1 Bootloader commands

Type ‘help or ‘?’ to see what kind of commands are there in bootloader.

```
Eddy> help
?        - alias for 'help'
base     - print or set address offset
cmp      - memory compare
coninfo - print console devices and information
cp       - memory copy
...
printenv- print environment variables
reset    - Perform RESET of the CPU
run      - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv   - set environment variables
tftpboot- boot image via network using TFTP protocol
version - print monitor version
Eddy>
```

- help : display list of commands

- install : Install firmware at bootloader level. Use this when the firmware is corrupted. For more information, refer to chapter 9.

- printenv : Display all the environmental variables. When used with the variable name, only its value is displayed. (example: printenv serverip)

- setenv : Set environmental variables. The format is “setenv [name of the environmental variable] [value]”. If you need to remove the value, use ‘setenv [name of the environmental variable]’ when you do specify the value, it will be deleted.

- saveenv : Save the modified value in Eddy. If you do not save it, modified value will not be applied.

- tftpboot : Used when you want to connect to Ethernet in bootloader to download the firmware image. For more information, refer to chapter 9. Appendix

6.2 Using Environmental Variable

When you use printenv command from the bootloader, environmental variables will be displayed as following.

```
Eddy〉 printenv
baudrate=115200
ethaddr=00:05:f4:00:90:57
ethact=macb0
wan=static
filesize=1E5070
fileaddr=20000000
gatewayip=192.168.0.254
netmask=255.255.255.0
ipaddr=192.168.0.212
serverip=192.168.0.220
dnsip=168.126.63.1
product=Eddy v2.5
firmware=2.5.3.1 rev ??
…
/flash/dk_default.zip=/
userinit=/flash/userinit
stdin=serial
stdout=serial
stderr=serial
Environment size: 955/65532 bytes
Eddy〉
```

Use setenv and saveenv commands to change the variables. Variables used here are same in Kernel therefore you can save and edit the value from bootloader or from Kernel.

For example, when you are trying to get the IP of WAN in Eddy, go to bootloader and check for the value for ipaddr or change the value and access through the web browser.

Additionally, you can change the ID and PW to connect to Eddy or use userinit variable to run the script you want to execute. Environmental variables such as /flash/dk_default.zip and /flash/eddy_wifi30.zip can apply file system to Eddy which are stored in /flash.

Environmental variables which name start with /flash is a system path that is trying to be uncompressed. For example, If you want to apply a user defined file system, userfs.zip, to Eddy, save userfs.zip in /flash and run setenv /flash/userinit/ command from bootloader. Use saveenv to save and reboot to see the user created userinit is applied in Eddy.

```
Eddy> setenv /flash/userfs.zip /
Eddy> saveenv
Saving Environment to dataflash...
```

Chapter 7.   Library

In this chapter, it will cover various APIs for users who wants to use Eddy DK to develop a program.

7.1   Before you start

All the functions introduced in this chapter is located in /userfs/sources/dk_serial/SB_APIs/ folder. They are APIs included in SB_*.c. Serial related exemplary programs are provided with Eddy DK will be using these so please refer to source code examples and Makefile.

7.1   Makefile

Library sources are saved as SB_*.c under /userfs/sources/dk_serial/SB_APIs/ folder. To use these libraries, you need to include it from Makefile, so refer to Makefile under userfs/sources/dk_serial folder.

7.2   System Family Functions

Timer and delay functions used when writing an application program.

SB_GetTick

Function	After Eddy boots return the time spent in msec.
Format	Unsigned long SB_GetTick (Void);
Parameter	None
Returns	0 ~ 4,294,967,295
Notice	Returned value is tick counter in msec, it will start again from 0 after max. value of 0xffffffff in unsigned long format. (It cycles in about 50 days.)

SB_msleep

Function	Delay by given time in msec.	
Format	void SB_msleep (int msec);	
Parameter	msec	Set time in msec
Returns	none	
Notice	Delay by given time in msec.	

SB_AliveTime

Function	Return the value in days, hours, minutes, and seconds since Eddy start operating.	
Format	void SB_AliveTime (int *day, int *hour, int *min, int *sec);	
Parameter	*day	Number of days operated (0 ~)
	*hour	hour (0 ~ 23)
	*min	minute (0 ~ 59)
	*sec	second (0 ~ 59)
Returns	None	
Notice		

7.3 Serial family function

Functions used to control UART.

SB_OpenSerial

Function	Open serial port	
Format	int SB_OpenSerial (int Port_No, int Interface);	
Parameter	Port_No	Serial port number 0 - 3 : First ~ fourth serial port
	Interface	Kinds of interface SB_RS232 : RS-232 interface SB_RS422 : RS-422 interface SB_RS485_NONE_ECHO : RS-485 none echo SB_RS485_ECHO : RS-485 echo
Returns	-1 ~ N	Handle opened serial ports -1 : Open error N : Handle opened serial port

Notice — Eddy provides up to 4 serial ports, but general models equipped with Eddy-CPU provides only 1 serial port.

DK board has 4 built-in serial ports so if you set it to recognize Eddy-CPU or Eddy DK, you can use all 4 ports.

.

SB_InitSerial

Function — Reset data communication style for serial port.

Format — Void SB_InitSerial (int Handle, char Speed, char LCR, char Flow);

Parameter — Handle — Handler for opened serial port

Speed

0	:	150 BPS,	1	:	300 BPS
2	:	600 BPS	3	:	1200 BPS:
4	:	2400 BPS	5	:	4800 BPS
6	:	9600 BPS	7	:	19200 BPS
8	:	38400 BPS	9	:	57600 BPS
10	:	115200 BPS	11	:	230400 BPS
12	:	460800 BPS	13	:	921600 BPS

LCR — X X P P S D D (8 bis binary)

P P : Parity Bits

0 0 : None, 0 1 : Odd, 1 0, 1 1: Even

S : Stop Bits

0 : 1 bits, 1 : 2 bits

D D : Data Bits

0 0 : 5 bits, 0 1 : 6 bits

1 0 : 7 bits, 1 1 : 8 bits

FlowControl — Type of flow control

0: no flow control

1: RTS/CTS flow control

2: Xon/Xoff flow contorl

Returns — None

Notice

SB_SendSerial

Function	Print data through serial port.	
Format	Void SB_SendSerial (int handle, char *data, int length);	
Parameter	handle	Handler for opened serial port
	data	Pointer for data to be printed
	length	Length of data to be printed
Returns	None	
Notice	When transmitting buffer is full, it will try 10 times with about 20 msec delay and return after displaying.	

SB_ReadSerial

Function	Read data from serial port	
Format	int SB_ReadSerial (int handle, char *data, int length, int wait_msec);	
Parameter	handle	Handle opened serial port
	data	Buffer pointer to be read
	length	Size of memory buffer (Length)
	wait_msec	After reading the received buffer, time delay until the next data to be received.
Returns	0 ~ n	Length of data to be read.
Notice	When wait_msec is set to 0, data from receiving buffer are read. If it is set to the value over 0, it will read the input data from the receiving buffer and wait until given time in msec, then read the data from the serial port as one packet. The maximum data size that it can read is by the buffer size in length. Wait_msec uses the value result of SB_GetDelaySerial function or by directly calculating the value.	

SB_GetMsr

Function	Read MSR register from serial port.	
Format	Char SB_GetMsr (int handle);	
Parameter	handle	Handler for opened serial port
Returns	Value	MSR Register value
		Bit 7 6 5 4 3 2 1 0
		Bit0: CTS change
		Bit1: DSR change
		Bit2: RI change
		Bit3: DCD change
		Bit4: CTS (0:Low, 1:High)
		Bit5: DSR (0:Low, 1:High)
		Bit6: RI (0:Low, 1:High)
		Bit7: DCD (0:Low, 1:High)
Notice		

SB_SetRts

Function	Controls RTS signal line in serial port.	
Format	Void SB_SetRts (int handle, int value);	
	handle	Handler for opened serial port
Parameter	Value	0: off Sets RTS signal to low.
		1: on Sets RTS signal to high.
Returns	None	
Notice		

SB_SetDtr

Function	Controls DTR signal in serial port.	
Format	Void SB_SetDtr (int handle, int value);	
	handle	Handler for opened serial port
Parameter	Value	0: off Sets DTR signal to low.
		1: on Sets DTR signal to high.
Returns	None	
Notice		

7.4 Ethernet Family Function

Input/output if current Network Eddy is using. This function is optimized socket API in Eddy, the users can use POSIX compatible standard socket API instead of this functions to develop.

SB_GetIp

Function	Read assigned IP address for Eddy.	
Format	Unsigned int SB_GetIp (char *interface);	
Parameter	Interface	Name of the Network Interface Set WAN port to “eth0” and LAN port to “eth1”.
Returns	Unsigned int	Return address in Unsigned int type.

Notice	Get IP address in used, not the IP address set in Eddy. When Eddy is in DHCP Client mode, it will get the Network IP address from DHCP Server.
	To convert IP address to string, refer to the following.
	struct in_addr addr;
	addr.s_addr = SB_GetIp ();
	printf ("IP Address : %s ", inet_ntoa(addr));	

SB_GetMask

Function	Read Subnet Mask address assigned to Eddy.	
Format	Unsigned int SB_GetMack (char *interface);	
Parameter	Interface	Name of the interface trying to read from Set WAN port to “eth0” and LAN port to “eth1”.
Returns	Unsigned int	Return Mask address in unsigned int type.
Notice	Refer to SB_GetIp	

SB_GetGateway

Function	Read Gateway address assigned to Eddy.	
Format	Unsigned int SB_SetGeteway(void);	
Parameter	None	
Returns	Unsinged int	Return Gateway address in unsigned int type.
Notice	Refer to SB_GetIp	

SB_ConnectTcp

Function	Connect to assigned server with TCP socket.	
Format	Int SB_ConnectTcp (char *IP_Address, int Socket_No, int Wait_Sec, Int Tx_Size, int Rx_Size);	
Parameter	IP_Address	IP address of the server to connect (in String)
	Socket_No	Socket number of the server to connect
	Wait_Sec	Connection standby time (in second)
	Tx_Size	Buffer size of Tx (in K bytes)
	Rx_Size	Buffer size of Rx (in K bytes)
Returns	-1 ~ N	Handle number for connected socket
		-1 : Connection fail
		N : Connected handle number
Notice	When immediate connection fails, wait by given seconds until the next attempt. Standby and return. Tx,Rx_Size is socket buffer size that can be set to 1 ~ 64. When the value less than 1 is given it will set as 4 Kbytes as default, and when it is larger than 64, it will be set to 64 Kbytes.	

SB_ListenTcp

Function	Request connection through TCP socket and standby.	
Format	Int SB_ListenTcp (int Socket_No, Int Tx_Size, int Rx_Size);	
Parameter	Socket_No	TCP socket number for standing by
	Tx_Bytes	Tx buffer size in socket (in K bytes)
	Rx_Bytes	Rx buffer size in socket (in K bytes)

Returns -1 ~ N Socket handle number for TCP connection

-1 : Socket connection standby fail

N : Socket handle number for TCP connection

Notice This is a Non-blocking function which returns without standing by when requesting for connection. Waiting for connection is processed at SB_AcceptTcp.

Tx,Rx_Size is socket buffer size that can be set to 1 ~ 64.

When the value less than 1 is given it will set as 4 Kbytes as default, and when it is larger than 64, it will be set to 64 Kbytes.

SB_AcceptTcp

Function Standby for TCP socket handler Network connection.

Format Int SB_AcceptTcp (int Socket_No, int wait_msec);

Parameter Socket_No TCP socket handle number waiting for connection

(Return value from SB_ListenTcp)

wait_msec Standby time (in msec)

Returns -1 ~ N New handle number connect to TCP socket.

-1: Socket error

0: Standby for connection

N: New handle number connect to TCP socket

Notice After connection, when new handle number is assigned, standing by previous handler close in this function.

SB_AcceptTcpMulti

Function	Allow multiple Network connections for TCP socket handlers standing by for connection.	
Format	Int SB_AcceptTcpMulti (int Socket_No, int wait_msec);	
Parameter	Socket_No	TCP socket handle number waiting for connection (Return value from SB_ListenTcp)
	wait_msec	Standby time (in msec)
Returns	-1 ~ N	New handle number connect to TCP socket. -1 : Socket error 0 : Standby for connection N : New handle number connect to TCP socket
Notice	After connection, when new handle number is assigned, standing by previous handler is not closed so max. 1024 sockets are allowed.	

SB_ReadTcp

Function	Read data from connected TCP socket.	
Format	Int SB_ReadTcp (int Handle, char *Buffer, int Buffer_Size);	
Parameter	Handle	Handle number connected to TCP socket
	Buffer	Buffer point where retrieved data will be stored
	Buffer_Size	Buffer to be saved
Returns	-1 ~ N	Length of data read -1 : Socket error 0 : No data read N : Length of data read
Notice	If return code is -1, connected device is disconnected, TCP socket handler should be closed.	

SB_CloseTcp

Function	Close TCP socket handle.	
Format	Int SB_CloseTcp (int Handle);	
Parameter	Handle	TCP socket handle number to be closed
Returns	None	
Notice	Close communication by shutting down the socket handle.	

SB_BindUdp

Function	Bind UDP socket	
Format	Int SB_BindUdp (int Socket_No);	
Parameter	Socket_No	UDP socket number to be bind
Returns	Handle	IDP socket handle number that is bound
		-1: Bind fail
		N : Bound UDP socket handle number
Notice		

SB_ReadUdp

Function	Read transmitted data from UDP socket bound to Network.	
Format	Int SB_ReadUdp (int Handle, char *Buffer, int Buffer_Size);	
Parameter	Handle	Handle number bound to UDP socket
	Buffer	Buffer point where retrieved data will be stored
	Buffer_Size	Size of buffer to be stored
Returns	-1 ~ N	Length of data read
		-1 : Socket error
		0 : No data read
		N : Length of data read
Notice	This function, when data is sent from UDP socket bound Network, remember the connected IP address and socket number and use it at SB_SendUdpServer.	

SB_SendUdpServer

Function	Transmit data by UDP socket (Server mode)	
Format	Int SB_SendUdpServer (int Handle, char *Buffer, int Data_Size);	
Parameter	Handle	Handle number bound to UDP socket
	Buffer	Buffer point where data will be transmitted
	Data_Size	Size of data to be transmitted
Returns	None	
Notice	This function, bound to UDP socket in Eddy, will send the data first to get the Network information about connected device. If you need to transmit the data first, use SB_SendUdpClient.	

SB_SendUdpClient

Function Transmit data through UDP socket. (Client mode)

Format Int SB_SendUdpClient (int Handle, char *Buffer, int Data_Size, Char *IP_Address, int Socket_No);

Parameter

Handle	Handle number bound to UDP socket
Buffer	Buffer point where data to be sent is stored
Data_Size	Length of data to be sent
IP_Address	IP address of the target where data will be reached.
Socket_No	Socket number of the target where data will be reached.

Returns None

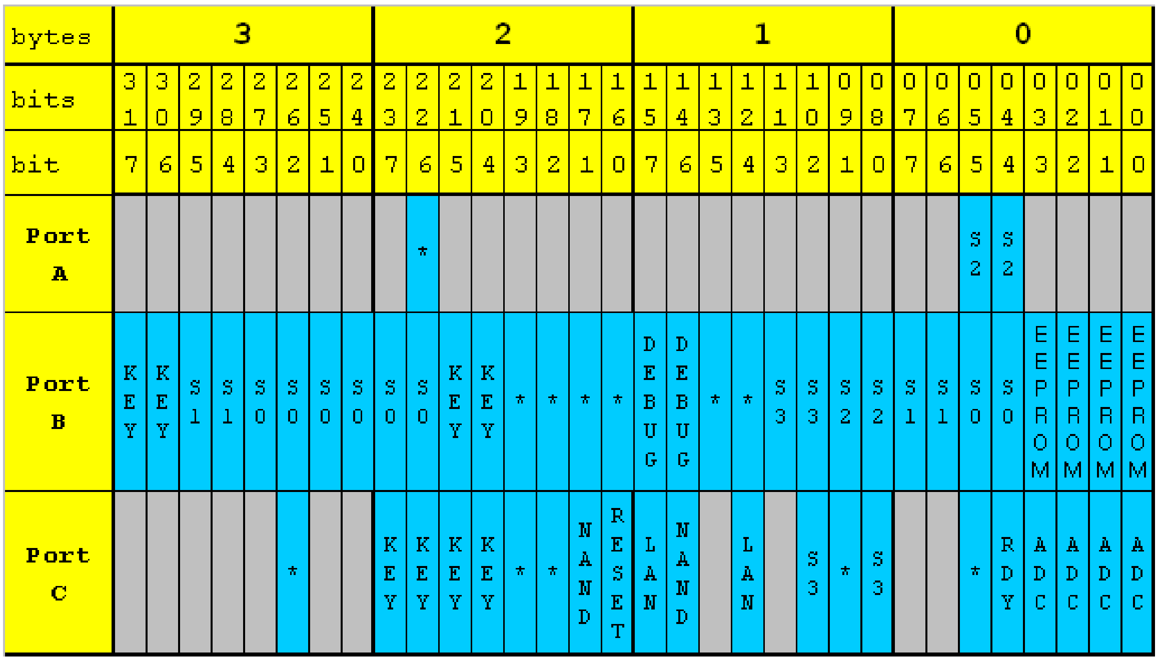
Notice This function can be used when UDP socket knows where to send the data.

To send the data first, use SB_SendUdpClient function.

7.5 GPIO Ioctl Function

It is a function to control GPIO port in Eddy-CPU (max. 56) and Eddy-S4M (max. 34).
By using GPIO separate port, you can detect the voltage of 3.3V or control the output. Pins provided in Eddy can be used to control other devices, but not used for GPIO only. Eddy has port A, B, and C groups where it provides 32 numbers of signals each.
Each ports in A, B and C in Eddy can be used for devices and for GPIOs. Basically, Eddy can be set from Web. For more usage, refer to “dk_test.c” sample source code in /userfs/sources/dk_test.

GPIO schema in Eddy-CPU

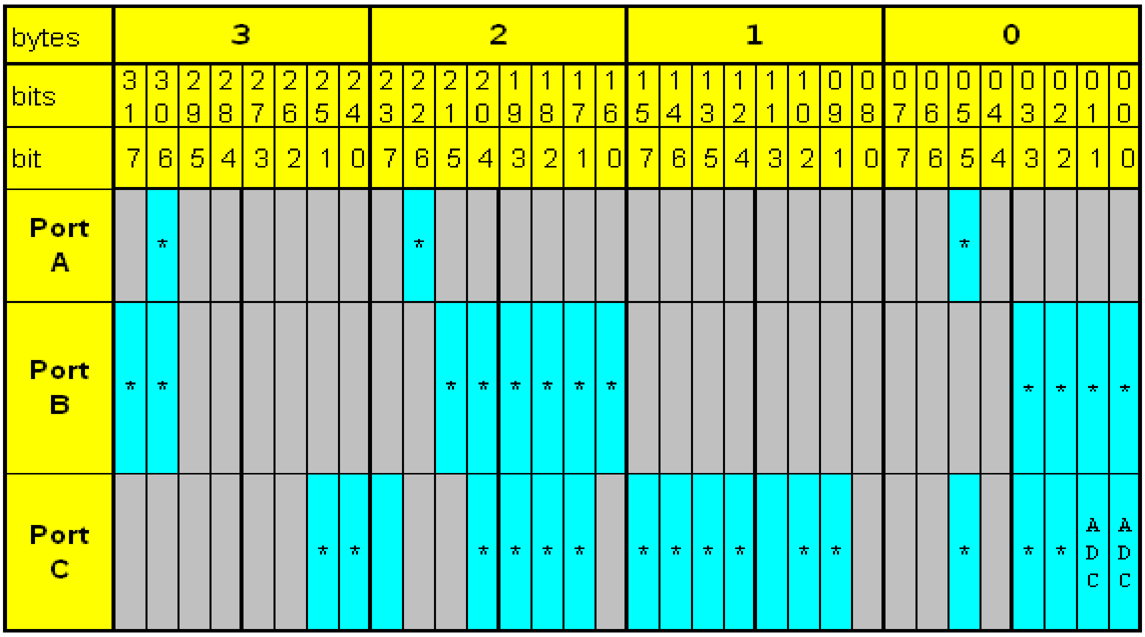


bytes	3								2								1								0							
bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Port A										*																	S2	S2				
Port B	KEY	KEY	S1	S1	S0	S0	S0	S0	S0	S0	KEY	KEY	*	*	*	*	DEBUG	DEBUG	*	*	S3	S3	S2	S2	S1	S1	S0	S0	EEPROM	EEPROM	EEPROM	EEPROM
Port C						*			KEY	KEY	KEY	KEY	*	*	NAND	RESET	LAN	NAND		LAN		S3	*	S3			*	RDY	ADC	ADC	ADC	ADC

From the chart above, when not used, parts in blue can be used as GPIO ports, but for grayed area, they are used by system and cannot be used.

	Description	Number of GPIOs
S0 ~ S3	Serial Port 1 ~ 4	20
Debug	Debug Port	2
Reset	Reset	1
Rdy	Ready LED	1
ADC	Analog Digital Converter	4
LAN	LAN Port	2
EEPROM	SPI (EEPROM)	4
NAND	NAND Flash	2
KEY	Key Pad	8
*	GPIO & User Peripheral	12

GPIO schema in Eddy-S4M



bytes	3								2								1								0							
bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Port A		*								*																	*					
Port B	*	*									*	*	*	*	*	*													*	*	*	*
Port C							*	*				*	*	*	*		*	*	*	*		*	*				*		*	*	ADC	ADC

From the chart above, when not used, parts in blue can be used as GPIO ports, but for grayed area, they are used by system and cannot be used.

	Description	Number of GPIO ports
ADC	Analog Digital Converter	2
*	GPIO & User Peripheral	32

Each port A, B, and C can express 32 numbers of GPIO, where in program, each bit represents each GPIO ports in int type 4 byte variable.

```
struct   eddy_gpio   {
        Unsigned   int   value    [3];                          // In/output status value by GPIO channel in each Port A, B, C
        Unsigned   int   mode     [3];             // Port A, B, C set in/output settings for each relative channel in GPIO
      Unsigned   int   pullup   [3];// Port A, B, C when setting GPIO channel for each GPIO
                                                          // set pullup or pulldown
        Unsigned   int   enable   [3];             // Port A, B, C whether each GPIO channel is used
};
```

For enable when bit is 0 → disable (Do not use as GPIO), 1 → Enable (Use as GPIO)
For mode when bit is 0 → Set to input mode, 1 → Set to output mode
For value when bit is 0 → In/output status is low, 1 → High
For pullup when bit is 0 → pulldown, 1 → pullup

SETGPIOINIT

Function After Eddy booted, initialize GPIOs used by system.

Format void ioctl (int fd, SETGPIOINIT, struct *gpio_struct);

Parameter

fd Handler number that opened GPIO device("/dev/eddy_gpio")

gpio_struct GPIO setting file registered from WEB configuration, struct pointer table for GPIO values in /etc/eddy_gpio.cfg file.

```
struct gpio_struct    {
        unsigned int value[3];
        unsigned int mode[3];
        unsigned int pullup[3];
        unsigned int enable[3];    };
```

Returns None

Notice Eddy-CPU provides total 56 numbers of GPIO, but for Eddy-S4M- max. 34 ports.

For Eddy-CPU, 56 ports are when you are using WAN only. If you use serial port, ADC, Rese, RDY LED, and other devices, numbers of GPIOs will decrease.

For Eddy-S4M, ADC is one of the device that uses 34 ports publicly.

This command initializes so that devices registered in Pinetd.c after Eddy boots can use the rest of ports in GPIO. Users do not have to use this command, but the inevitable need to be extremely careful in the use.

For example, set web configuration to use serial ports and reboot Eddy to use the corresponding port to be used as serial port and not GPIO port. With this command, it will allow you to use it as GPIO port where applications using serial ports will malfunction.

SETGPIOMOD_LM

Function	Set in a batch the configuration of port A, B, C.	
Format	void ioctl (int fd, SETGPIOMOD_LM, int *mode[3]);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	mode	Buffer pointer that the value of mode for Port A.B.C is saved. When 0 Bit then input, but if it is 1 then output.
Returns	None	
Notice	Bit that enables GPIOs to be used only matter. When you set other bits to any value they will not effect this.	

GETGPIOMOD_LM

Function	Read in a batch the direction of in/output of port A, B, C.	
Format	void ioctl (int fd, GETGPIOMOD_LM, int *mode[3]);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	mode	Buffer pointer that the value of mode for port A.B.C will be saved.
Returns	None	
Notice		

SETGPIOVAL_LM

Function	If the mode of all port A, B, C is set to output, set the output value in a batch.	
Format	void ioctl (int fd, SETGPIOVAL_LM, int *value[3]);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	value	Buffer pointer that the value of mode for port A.B.C is saved. If the bit value is 0 then low, else if 1 then high
Returns	None	
Notice	Bit that enables GPIOs to be used only matter. When you set other bits to any value they will not effect this.	

GETGPIOVAL_LM

Function	Read in a batch the configuration of port A, B, C.	
Format	void ioctl (int fd, GETGPIOVAL_LM, int *value[3]);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	value	Buffer pointer that the value of mode for port A.B.C is saved
Returns	None	
Notice		

SETGPIOPUL_LM

Function	If the mode of all port A, B, C is set to input, set the pullup status in a batch.	
Format	void ioctl (int fd, SETGPIOPUL_LM, int *pullup[3]);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	pullup	Buffer pointer that the value of pullup for port A.B.C is saved. If the bit value is 0 then pulldown, else if 1 then pullup.
Returns	None	
Notice	Bit that enables GPIOs to be used only matter. When you set other bits to any value they will not effect this.	

GETGPIOPUL_LM

Function	Read in a batch the pullup status of port A, B, C.	
Format	void ioctl (int fd, GETGPIOPUL_LM, int *pullup[3]);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	pullup	Buffer pointer that will save the pullup value of port A.B.C
Returns	None	
Notice		

SETGPIOMOD_LA

SETGPIOMOD_LB

SETGPIOMOD_LC

Function	Set the direction of in/output in one of the port in A, B, C.	
Format	void ioctl (int fd, SETGPIOMOD_L?, int *mode);	
Parameter	fd	Handle number that opened GPIO device ("/dev/eddy_gpio").
	mode	Buffer pointer that stores the mode value for the port If the bit value is 0 then input, else if 1 then output.
Returns	None	
Notice	Bit that enables GPIOs to be used only matter. When you set other bits to any value they will not effect this.	

GETGPIOMOD_LA

GETGPIOMOD_LB

GETGPIOMOD_LC

Function	Read the status of direction of in/output in one of the port in A, B, C.	
Format	void ioctl (int fd, GETGPIOMOD_L?, int *mode);	
Parameter	fd	Handle number that opened GPIO device ("/dev/eddy_gpio").
	mode	Buffer pointer that stores the mode value for the port
Returns	None	
Notice		

SETGPIOVAL_LA

SETGPIOVAL_LB

SETGPIOVAL_LC

Function	If the port is set for output, set the output value.	
Format	void ioctl (int fd, SETGPIOVAL_L?, int *value);	
Parameter	fd	Handle number that opened GPIO device ("/dev/eddy_gpio").
	Value	Buffer pointer that stores the mode value for the port If the bit value is 0 then low, else if 1 then high.
Returns	None	
Notice	Bit that enables GPIOs to be used only matter. When you set other bits to any	

value they will not effect this.

GETGPIOVAL_LA
GETGPIOVAL_LB
GETGPIOVAL_LC

Function	Read the status of direction of in/output in one of the port in A, B, C.	
Format	void ioctl (int fd, GETGPIOVAL_L?, int *value);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	value	Buffer pointer that stores the mode value for the port
Returns	None	
Notice		

SETGPIOPUL_LA
SETGPIOPUL_LB
SETGPIOPUL_LC

Function	Set the port to pullup if it is set to input.	
Format	void ioctl (int fd, SETGPIOPUL_L?, int *pullup);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	pullup	Buffer pointer that stores the pullup value for the port If the bit value is 0 then pulldown, else if 1 then pullup.
Returns	None	
Notice	Bit that enables GPIOs to be used only matter. When you set other bits to any value they will not effect this.	

GETGPIOPUL_LA
GETGPIOPUL_LB
GETGPIOPUL_LC

Function	Read the status of pullup of in/output in one of the port in A, B, C.	
Format	void ioctl (int fd, GETGPIOPUL_L?, int *pullup);	
Parameter	fd	Handle number that opened GPIO device (“/dev/eddy_gpio”).
	pullup	Buffer pointer that stores the pullup value for the port
Returns	None	
Notice		

7.6 ADC related Function

Eddy-CPU provides 4 ADC (Analog Digital Converter) channels. There are a temperature sensor and an ambient light sensor in Eddy DK where developers can get the real-time information from ADC. A sample program for ADC interface is Eddy_APPs/test_adc.c, by which developers can use it for developing a program.

ADCSETCHANNEL

Function Set whether to use or not the 4 channels in ADC device.

Format void ioctl (int fd, ADCSETCHANNEL, int *channel);

Parameter
fd Handle number that opened ADC device (“/dev/adc”).
channel Buffer pointer that contains whether to use or not about the channel.

Returns

Notice

```
x x x x x x x x (bits)
|   |   |   |------   channel   1 (temperature sensor)
|   |   |---------   channel   2 (illumination sensor)
|   |-------------- channel   3 (future use)
|----------------- channel   4 (future use)
```

ADCGETVALUE

Function Read the information about the operating status of 4 channels in ADC device.

Format void ioctl (int fd, ADCGETVALUE, struct adc_struct *channels);

Parameter
fd Handle number that opened ADC device (“/dev/adc”).
channels Buffer pointer that contains the operating status of 4 channels.

Returns

Notice

```
struct adc_value   {
            int ch1_value;
            int ch2_value;
            int ch3_value;
            int ch4_value;
};
```

Chapter 8. Modify the Web Page

Eddy provides CGI source code and HTML documents so that users can use them directly or modify and use them. These provided files are the files which actual Eddy DK uses. This chapter explains how apply those with sample source codes. Samples in this chapter are based on Windows 7, for Windows XP, they are similar or the same.

8.1 WEB Configuration

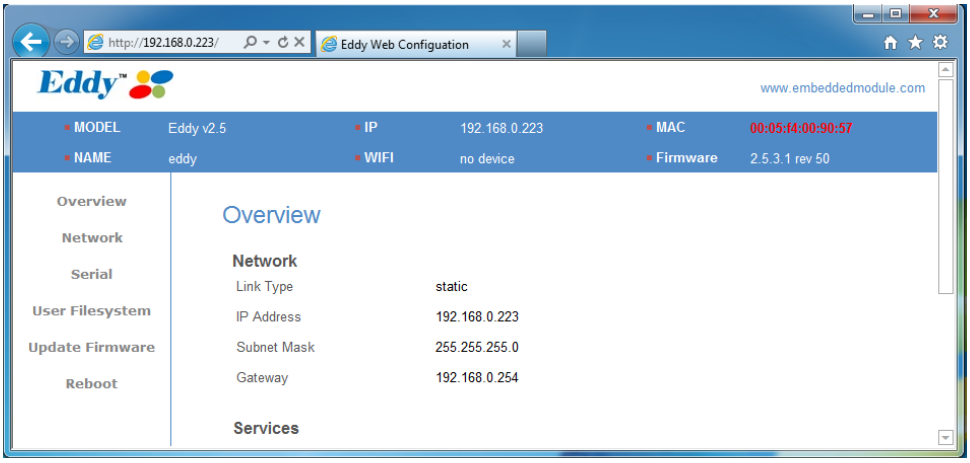
All executable source codes in Eddy is located in /usr/local/www ,
CGI source codes that you need information for html are located in /usr/local/www/cgi-bin/ .

8.2 How to write a sample code

First login to Eddy and move to /usr/local/www/menu folder. When you check the file contents in this directory, they are as follows.

```
# cd /usr/local/www/menu
# ls
00Overview.cgi            95Reboot.cgi              serial_head.inc
01Network.cgi             __Updating.html           userfs_list_head.inc
20Serial.cgi            network.inc            userfs_list_tail.inc
80User_Filesystem.cgi     overview.inc              userfs_upload.inc
90Update_Firmware.html   serial_body.inc
```

Documents in this folder are the exact menus that will be displayed when you try to connect Eddy through the Web browser. Commonly, they will have 2 digits of numbers in front. These numbers will be applied to the menus in order. The screenshot below shows the applied menu by the order.



MODEL Eddy v2.5
IP 192.168.0.223
MAC 00:05:f4:00:90:57
NAME eddy
WIFI no device
Firmware 2.5.3.1 rev 50

Overview

Network

Link Type static
IP Address 192.168.0.223
Subnet Mask 255.255.255.0
Gateway 192.168.0.254

Services

When the user save a temporary files under /usr/local/www/menu folder in Eddy and same them in the right format, the Web server will load the webpage and apply them. Detailed format is as follows.

XXMenuname.cgi

For the webpage, two digits of numbers used in the filename then the name that comes after the digits are displayed in the webpage in the menu. When the corresponding menu is selected, related content will be displayed in the middle and the top of the content will have the menus displayed automatically. File for the pages can be used with CGI or html format.

Following example is written under assuming that they are added after the user is logged in Eddy.

```
# cd /usr/local/www/menu
# vi 93System_Time.cgi
```

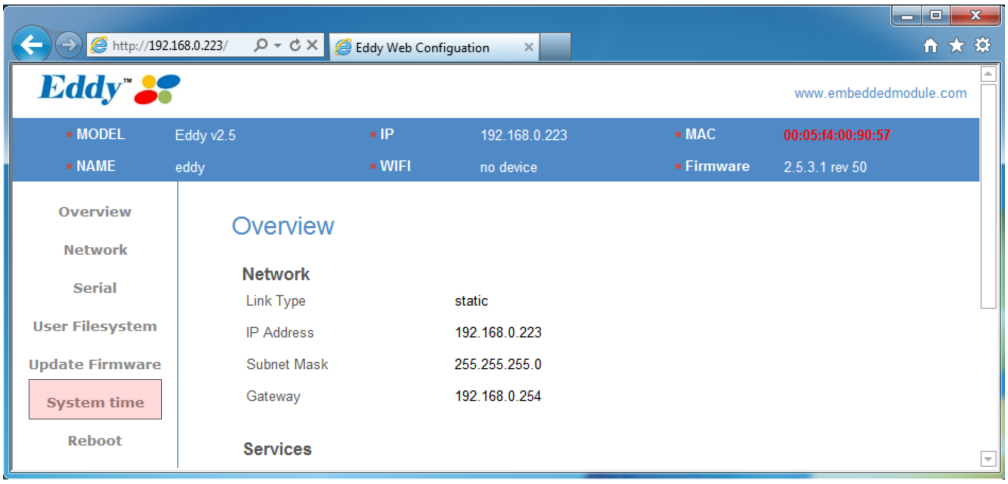
Use vi editor to add the following content.

```
#!/bin/sh
date
```

User vi editor to save the progress and quit. Check the name of the file.

```
# ls
00Overview.cgi          93System_Time.cgi      serial_body.inc
01Network.cgi           95Reboot.cgi           serial_head.inc
20Serial.cgi            __Updating.html        userfs_list_head.inc
80User_Filesystem.cgi   network.inc            userfs_list_tail.inc
90Update_Firmware.html  overview.inc           userfs_upload.inc
```

When you open the webpage or update it, you will see the added menu.



MODEL	Eddy v2.5	IP	192.168.0.223	MAC	00:05:f4:00:90:57
NAME	eddy	WIFI	no device	Firmware	2.5.3.1 rev 50

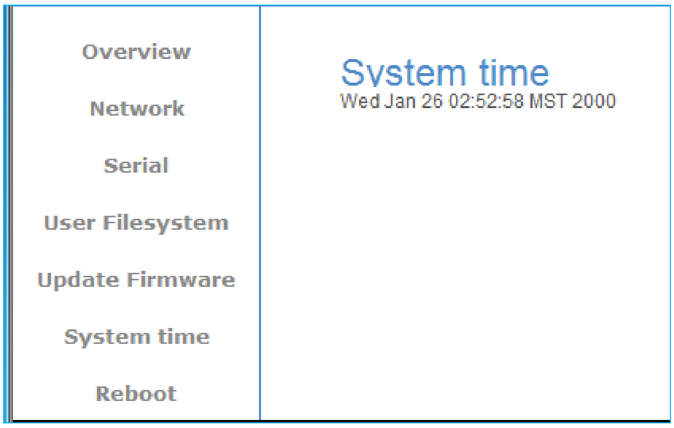
Overview

Network

Link Type	static
IP Address	192.168.0.223
Subnet Mask	255.255.255.0
Gateway	192.168.0.254

Services

When you choose the added menu, following page will be displayed.



System time

Wed Jan 26 02:52:58 MST 2000

When Eddy gets reset or powered down, these pages will be deleted. If you want to maintain the pages, you need to apply them to the system.

Move to the directory where Eddy SDK is located, /userfs/dk_default/usr/local/www/menu, and create or modify the webpage. When creating page is complete build the file system.

Build file system and refer to chapter 5 User File System to apply it to Eddy.

When above tasks are complete, login to Eddy and check /usr/local/www/menu folder for user created page or use the web browser to check the content.

.

Chapter 9. Appendix

This chapter explains how to recover when your flash in Eddy is corrupted or you cannot boot from it.

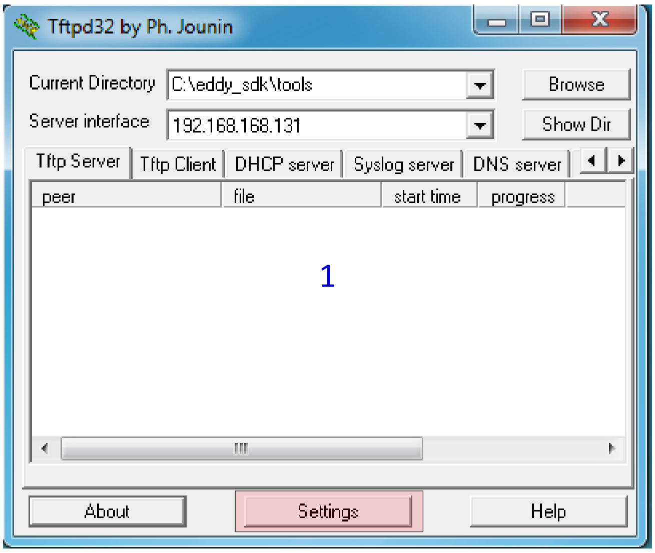
9.1 System recovery via bootloader

System booting will not be affected even if the user area in the flash is damaged. However, Eddy should be reset to factory default when there are fault in user program, system can reboot continuously or IP settings have wrong value stored that it will not connect to other devices. Before reverting it back to factory default, a firmware can be uploaded from boot loader. In order to do this, a computer with TFTP server built in Linux is needed.

> Caution:
> If the bootloader is damaged, it cannot be recovered by using bootloader. Therefore, do not use commands other than the one provided by the manual.

9.1.1 Install TFTP under Windows

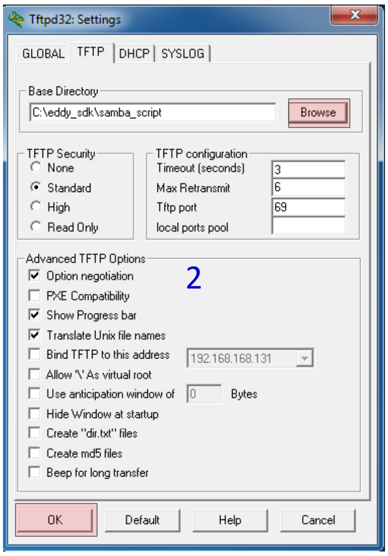
The following explains how to recover the system from bootloader in Windows XP SP3. If you have other OS, you would need a TFTP-server working on the system. When TFTP-server is installed in C:\eddy_sdk\ the execution file is tftpd32.exe located in C:\eddy_sdk\tools\. The following is the basic TFTP settings that you should check.



Tftpd32 by Ph. Jounin

Current Directory: C:\eddy_sdk\tools

Server interface: 192.168.168.131



Tftpd32: Settings

Base Directory: C:\eddy_sdk\samba_script

TFTP Security: ○ None ● Standard ○ High ○ Read Only

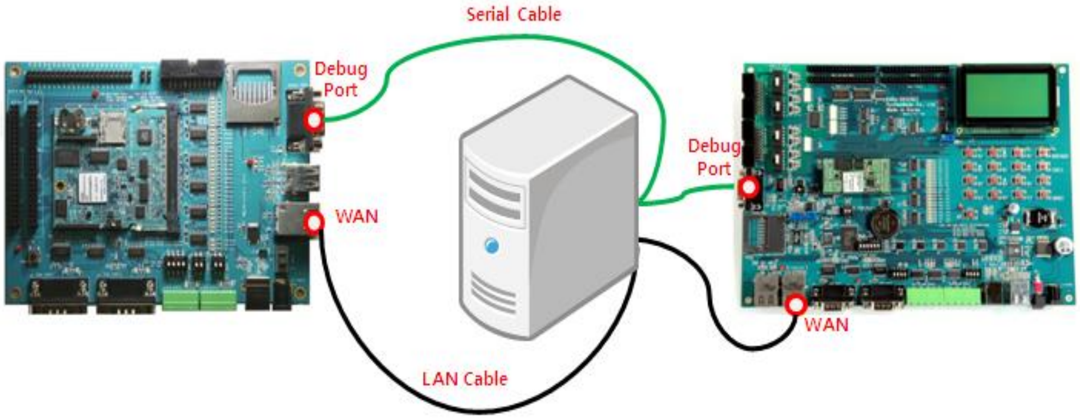
TFTP configuration: Timeout (seconds) 3; Max Retransmit 6; Tftp port 69; local ports pool

Advanced TFTP Options:
- ☑ Option negotiation
- □ PXE Compatibility
- ☑ Show Progress bar
- ☑ Translate Unix file names
- □ Bind TFTP to this address 192.168.168.131
- □ Allow '\' As virtual root
- □ Use anticipation window of 0 Bytes
- □ Hide Window at startup
- □ Create "dir.txt" files
- □ Create md5 files
- □ Beep for long transfer

When you execute tftpd32.exe a program will show a screen as displayed in 1 above. When you select settings in 1, configuration windows as shown in 2 will show. Use browse from 2 to select the location of the firmware file. After setting is complete, click OK to finish. This completes how to set TFTP server settings under Windows.

9.1.2 Install or recover hardware

Use the LAN Cable provided with Eddy DK and connect WAN port from DK with LAN port in the PC. With the serial cross cable provided with the product, connect a serial port in your PC with the debug port in the DK and use terminal to connect to serial port in the PC. Set the configuration for serial port in the computer to 115200 bps, 8 data bit, No parity, and 1 stop bit then supply power to DK board and press the reset button immediately for 3 seconds to enter bootloader.



[For Eddy-S4M DK] [For Eddy-CPU DK]

The following is a screenshot after entering bootloader.

```
RomBOOT



U-Boot 1.3.4-svn12 (Nov 15 2012 - 18:41:00)


DRAM:   32 MB
In:     serial
Out:    serial
Err:    serial
Net:    macb0
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Eddy>
```

Use commands in Bootloader to recover firmware image by copying it to flash memory. You must set the virtual IP address and IP address in TFTP server from bootloader in Eddy to use Kernel and firmware image file to recover. To check the current settings, use "printenv" command from bootloader to check the IP address of Eddy and TFTP server.

```
Eddy> printenv
baudrate=115200
…
ipaddr=192.168.0.223
serverip=192.168.0.220
dnsip=168.126.63.1
product=Eddy v2.5
firmware=2.5.3.1 rev 50
…
Environment size: 938/65532 bytes
Eddy>
```

Highlighted in blue images are information related to firmware. If these two information do not match, use following steps to synchronize them.

When you use more than one Eddy in same Network, the default IP will conflict with each other. To change the temporary IP address and IP address of TFTP server in Eddy, please refer to the example below.

```
Eddy> setenv ipaddr 192.168.0.212
Eddy> setenv serverip 192.168.0.89
Eddy> saveenv
```

When you check the IP information, start recovering.

install bootloader <bootloader firmware name> ; recover bootloader area

(Caution: When bootloader is damaged, recovery from DK board is not possible.)

install <firmware name>

When you execute as shown below, you can download the image file from configured TFTP server and recover.

```
Eddy> install eddy-2.5.3.1.bin
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Using macb0 device
TFTP from server 192.168.0.89; our IP address is 192.168.0.212
Filename 'eddy-2.5.3.1.bin'.
Load address: 0x20000000
Loading:
############################################################
#####

############################################################
#####

############################################################
#####

############################################################
#####

############################################################
```

After recovering process is completed, reboot your system.

```
Eddy>reset
```

9.1.3 Troubleshooting on recovery

```
Eddy> install eddy-2.5.3.1.bin
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Using macb0 device
TFTP from server 192.168.0.89; our IP address is
192.168.0.212
Filename 'eddy-2.5.3.1.bin'.
Load address: 0x20000000
```

If message as above is displayed but halts, check for WAN connection and IP address of PC where TFTP-server is installed. (Base on above example)

```
Eddy> install eddy-2.5.3.1.bin
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Using macb0 device
TFTP from server 192.168.0.89; our IP address is 192.168.0.212
Filename 'eddy-2.5.3.1.bin'.
Load address: 0x20000000
Loading: *
TFTP error: 'File not found' (1)
Starting again
```

When above message is displayed and halts, check for the firmware version or if the name is identical. Red colored from above must be same as the firmware installed in the TFTP-server in the PC.

```
Eddy> install eddy-2.5.3.1.bin
macb0: link up, 100Mbps half-duplex (lpa: 0x00e0)
Using macb0 device
TFTP from server 192.168.0.89; our IP address is 192.168.0.212
Filename 'eddy-2.5.3.1.bin'.
Load address: 0x20000000
Loading: #T T T T T T T T T##T
```

When above message is displayed, there are same MAC address or IP address in the same Network. This case check if there are same Eddy product in the same Network.

9.2 System recovery via port

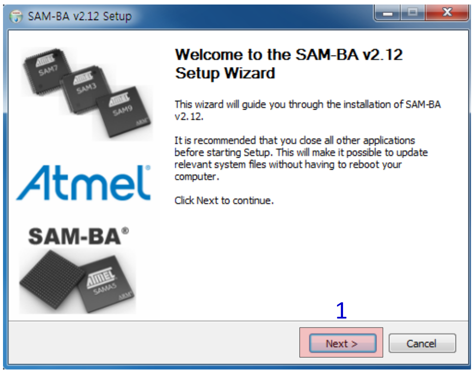
System booting will not be affected even if the user area in the flash is damaged. However, Eddy should be reset to factory default when there are fault in user program, system can reboot continuously or IP settings have wrong value stored that it will not connect to other devices. This section explains how to use USB to upload the firmware and revert it back to factory default.

(Caution: USB system recovery may be influenced by the characteristics of the USB port in the PC.)

9.2.1 Prepare to recover USB system

Make a temporary folder (for example, C:\SystemBase\USB_recovery) and extract files in there from Eddy-CPU_v25_USB_Recovery.zip in SDK\Windows\USB_ recovery folder. The files are included in the Eddy DK CD.

Install USB Tool program. You can start installing by double clicking Sam-ba_2.12.exe file.

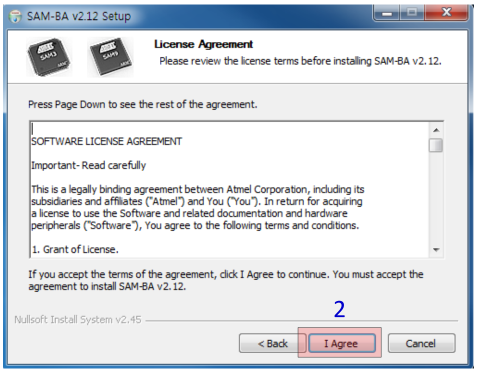


Welcome to the SAM-BA v2.12 Setup Wizard

This wizard will guide you through the installation of SAM-BA v2.12.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



License Agreement

Please review the license terms before installing SAM-BA v2.12.

Press Page Down to see the rest of the agreement.

SOFTWARE LICENSE AGREEMENT

Important- Read carefully

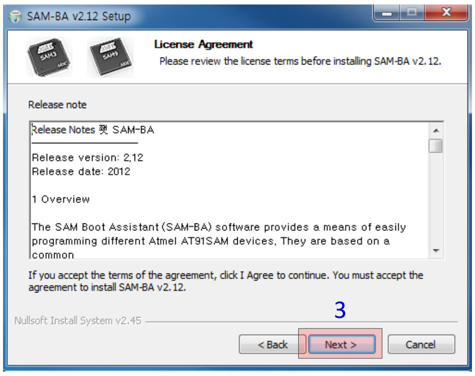
This is a legally binding agreement between Atmel Corporation, including its subsidiaries and affiliates ("Atmel") and You ("You"). In return for acquiring a license to use the Software and related documentation and hardware peripherals ("Software"), You agree to the following terms and conditions.

1. Grant of License.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install SAM-BA v2.12.

- 1. Click “Next”

- 2. Click “I Agree”



SAM-BA v2.12 Setup

License Agreement

Please review the license terms before installing SAM-BA v2.12.

Release note

Release Notes 꽷 SAM-BA

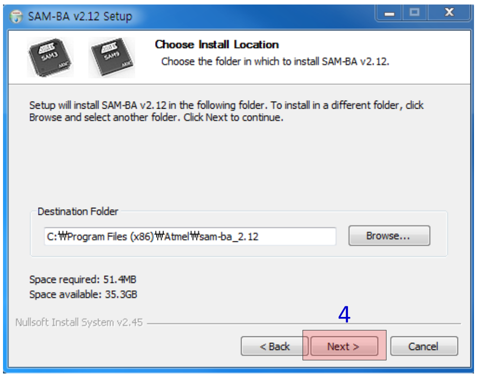
Release version: 2.12
Release date: 2012

1 Overview

The SAM Boot Assistant (SAM-BA) software provides a means of easily programming different Atmel AT91SAM devices. They are based on a common

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install SAM-BA v2.12.

Nullsoft Install System v2.45



SAM-BA v2.12 Setup

Choose Install Location

Choose the folder in which to install SAM-BA v2.12.

Setup will install SAM-BA v2.12 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

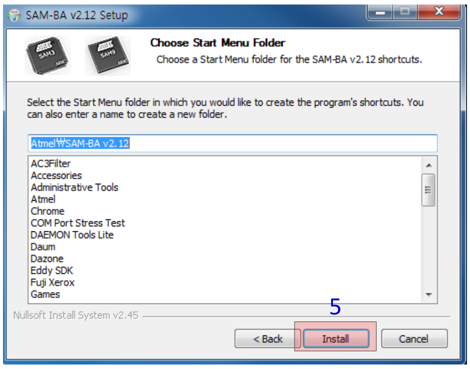
Destination Folder: C:₩Program Files (x86)₩Atmel₩sam-ba_2.12

Space required: 51.4MB
Space available: 35.3GB

Nullsoft Install System v2.45

- 3. Click “Next”

-4. The default location of installation can be changed if you want. Click “Next” to proceed.



SAM-BA v2.12 Setup

Choose Start Menu Folder

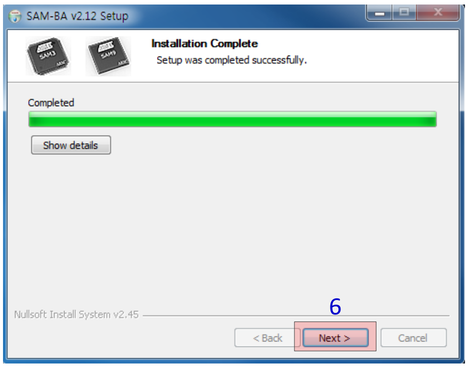
Choose a Start Menu folder for the SAM-BA v2.12 shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Atmel₩SAM-BA v2.12

AC3Filter
Accessories
Administrative Tools
Atmel
Chrome
COM Port Stress Test
DAEMON Tools Lite
Daum
Dazone
Eddy SDK
Fuji Xerox
Games

Nullsoft Install System v2.45



SAM-BA v2.12 Setup

Installation Complete

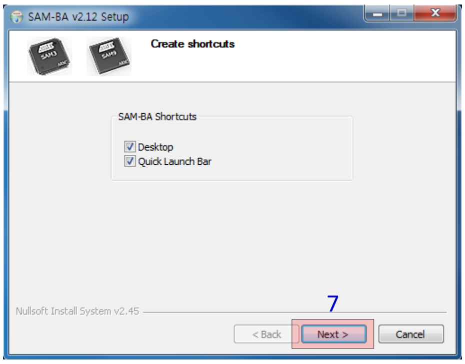
Setup was completed successfully.

Completed

Nullsoft Install System v2.45

- 5. Click “Install”

- 6. After copying the files is complete, click “Next”.



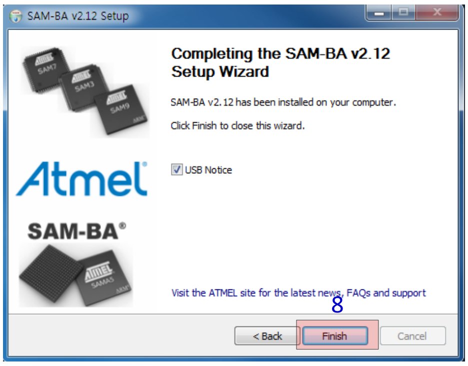
SAM-BA v2.12 Setup

Create shortcuts

SAM-BA Shortcuts

☑ Desktop
☑ Quick Launch Bar

Nullsoft Install System v2.45



SAM-BA v2.12 Setup

Completing the SAM-BA v2.12 Setup Wizard

SAM-BA v2.12 has been installed on your computer.

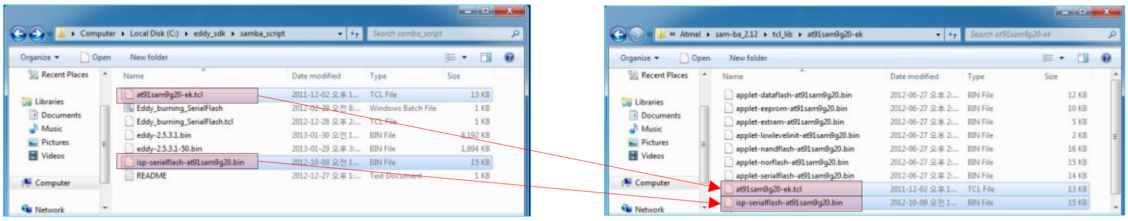
Click Finish to close this wizard.

☑ USB Notice

Visit the ATMEL site for the latest news, FAQs and support

- 7. If you want to add a shortcut click on the check box near “Desktop” or “Quick Launch Bar” and click “Next”.

- 8. Click “Finish” to complete the installation.



Computer ▸ Local Disk (C:) ▸ eddy_sdk ▸ samba_script

Name	Date modified	Type	Size
at91sam9g20-ek.tcl	2011-12-02 오후 1...	TCL File	13 KB
Eddy_burning_SerialFlash	2012-02-29 오전 8:...	Windows Batch File	1 KB
Eddy_burning_SerialFlash.tcl	2012-12-28 오후 2:...	TCL File	1 KB
eddy-2.5.3.1.bin	2013-01-30 오전 1...	BIN File	8,192 KB
eddy-2.5.3.1-50.bin	2013-01-29 오후 3:...	BIN File	1,894 KB
isp-serialflash-at91sam9g20.bin	2012-10-09 오전 1...	BIN File	15 KB
README	2012-12-27 오후 1:...	Text Document	1 KB

« Atmel ▸ sam-ba_2.12 ▸ tcl_lib ▸ at91sam9g20-ek

Name	Date modified	Type	Size
applet-dataflash-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	12 KB
applet-eeprom-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	10 KB
applet-extram-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	5 KB
applet-lowlevelinit-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	2 KB
applet-nandflash-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	16 KB
applet-norflash-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	15 KB
applet-serialflash-at91sam9g20.bin	2012-06-27 오후 2:...	BIN File	14 KB
at91sam9g20-ek.tcl	2011-12-02 오후 1...	TCL File	13 KB
isp-serialflash-at91sam9g20.bin	2012-10-09 오전 1...	BIN File	15 KB

-

Move from the folder where SDK is installed to samba_script folder. Copy at91sam9g20-ek.tcl and isp-serialflash-at91sam9g20.bin files to C:\Program Files\Atmel\sam-ba_2.12\tcl_lib\at91sam9g20-ek and over write them.

– Look for the file under firmware where you extracted the zip file.
"eddy-2.5.3.1.bin"

- Eddy_burning_SerialFlash.bat: This file executes TCL file to upgrade and create a log file after it uploads firmware through USB to DK board. Make sure as shown below that name of eddy-2.5.3.1.bin file and the name of Eddy_burning_SerialFlash.tcl file are the same as the file you downloaded.

```
sam-ba.exe \usb\ARM0 AT91SAM9260-EK Eddy_burning_SerialFlash.tcl > logfile.log
notepad logfile.log
```

Eddy_burning_SerialFlash.tcl: transplants the firmware file to the board.

```
…
################################################################################
#   Main script: Load the linux demo in SerialFlash,
#                    Update the environment variables
################################################################################
array set df_mapping {
     kernelFileName          "eddy-2.5.3.1.bin"

     baseAddr                0xD0000000
     kernelOff               0x00000000
}
```

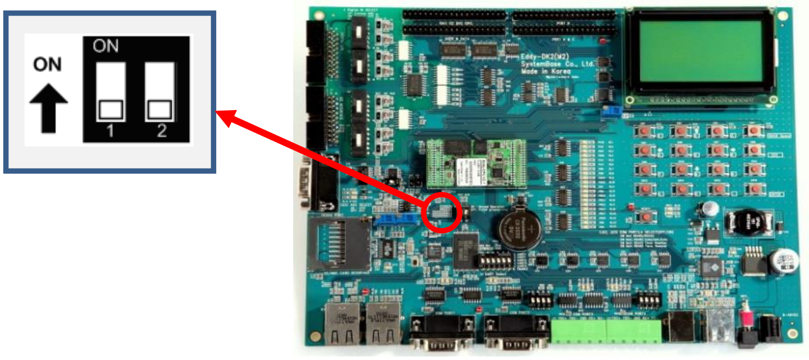
9.2.2 Install driver for DK board

To make your PC to recognize Eddy DK and Eddy-S4M-DK from a USB device, install the driver in following order.

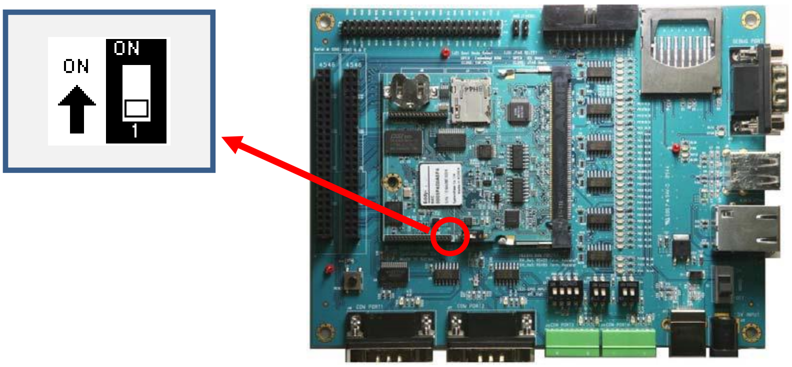
Turn off DK board.

Connect USB cable from DK to the PC.

Set USB to standby mode by setting dip switch, as shown below, to off.



[For Eddy-CPU DK, dip switch S6]



[For Eddy-S4M DK, dip switch S1]

Turn on DK board.

When the PC recognize the DK board, to install a new driver, a dialog box is created.

Select “Yes, connect when the device is plugged in.” then “Next”.

Select “Automatically install the software” then “Next”.

Select the folder where the driver is installed and select atm6124.sys ATMEL AT91xxxxx Test Board driver and then “C” to continue.

Select “Finish” to complete the installation.

Turn on the dip switch on the DK board.

When DK board is not recognized from the PC even after the driver is installed as shown below.
- Check for conflicts with VMware or any programs that use virtual device driver.
- When recovering from USB, stop all the programs that uses virtual device driver.

What to do when the driver automatically installs but with the different name.

Check with device manager and see if "AT91 USB to Serial Converter" is installed. If not, remove the driver.

If you off/on the USB connection or the power, the PC will search for the driver again. If the driver installation window automatically runs, click "Skip download for Windows Update software driver" to cancel the installation.

Go back to the device manager and right mouse click on Eddy which shows as Unknown Device to start the software update for the driver. Automatic/Manual driver install selection window will appear. Select "Look for software driver from the computer."

Click on the search button in [Search for driver software in the following locations] and go to the folder where the driver file is located. Click "Next" but when you see a warning pop-up window, ignore it and install the driver.

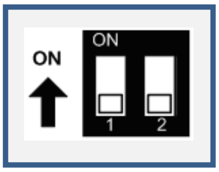
Check for the name of the installed driver before you close the windows.

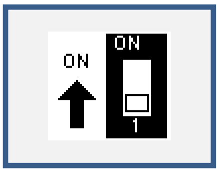
9.2.3 Execute USB system recovery

Turn off the power in the DK board.

Connect USB cable from DK board to the PC.

Set it to USB standby mode by turning off the dip switch in the DK board.

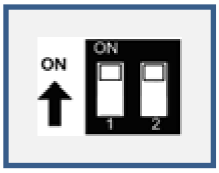


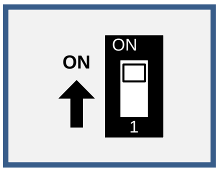


[Eddy-CPU DK:S6] [Eddy-S4M DK:S1]

Turn on the power in the DK board.

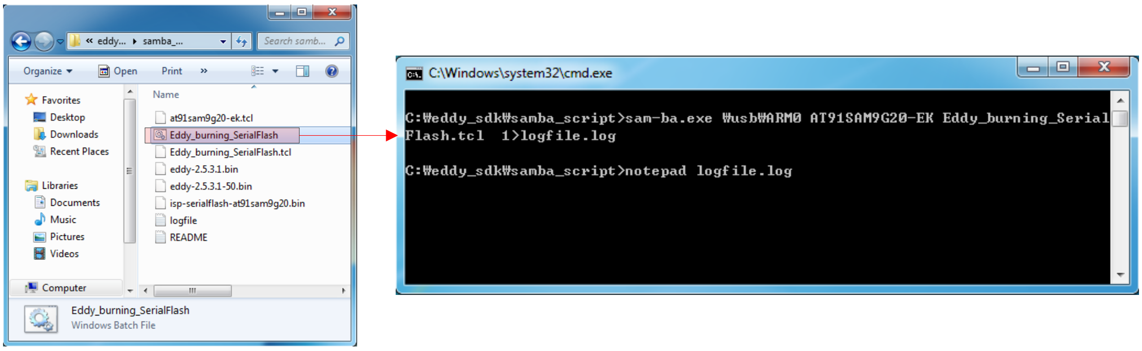
After about 5 seconds later, turn on the dip switch in the DK board so that the data can be written in flash.





[Eddy-CPU DK:S6] [Eddy-S4M DK:S1]

In the folder where USB system recovery is installed, run Eddy_burning_SerialFlash.bat to start the USB recovery. After this batch file is executed, you can check the result from logfile.log file after few moments.



```
C:₩eddy_sdk₩samba_script>sam-ba.exe ₩usb₩ARM0 AT91SAM9G20-EK Eddy_burning_Serial
Flash.tcl  1>logfile.log

C:₩eddy_sdk₩samba_script>notepad logfile.log
```

After the task is complete, logfile.log will be opened automatically where you can find the following message in it.

```
…
-I- Complete 99%
-I-        Writing: 0x1000 bytes at 0x7FF000 (buffer addr : 0x302000)
-I-        0x1000 bytes written by applet
```

After upgrading is complete, you can use logfile.log to check whether USB system recovery was successful. Reset the power in DK board to check the system operates normally after USB system recovery.

If log message shows that it was not successful, refer to “Troubleshoot USB System Recovery”.

9.2.4 Troubleshooting USB system recovery

If USB recovery file name is incorrect following message will be displayed. In this case, check from Eddy_burning_xxxxFlash.bat file or Eddy_burning_xxxxFlash.tcl file whether the firmware file name and the downloaded file name is the same. If not, change the name and reinstall.

```
…

-E- Script File Eddy_burning_SerialFlash.tcl returned error : could not read "eddy-2.5.3.1.bin": no such file or directory - could not read "eddy-2.5.3.1.bin": no such file or directory
```

If a log error shown as below is displayed, check the USB cable that connects from PC to the DK board and reinstall it.

```
…
-E- Connection \usb\ARM0 not found
-E- Connection list : COM2 COM3 COM4 COM5
```

If a following error message is displayed, check S6 dip switch in Eddy DK board whether it is turned down. Turn it up and reinstall.

```
…
-E- Script File Eddy_burning_SerialFlash.tcl returned error : Can't detect known device - Can't detect known device
```

If a following error message is displayed, refer to Prepare USB recovery and copy at91sam9g20-ek.tcl and isp-serialflash-at91sam9g20.bin files.
Then, overwrite them at C:\Program Files\Atmel\sam-ba_2.12\tcl_lib\at91sam9g20-ek.

```
…
-I- Loading applet applet-lowlevelinit-at91sam9g20.bin at address 0x200000
-I- Memory Size : 0x0 bytes
-I- Buffer address : 0x4
-I- Buffer size: 0x0 bytes
-I- Applet initialization done
-I- Low level initialized
-I- External RAM Settings :  extRamVdd=1, extRamType=0, extRamDataBusWidth=32, extDDRamModel=0
-I- Loading applet applet-extram-at91sam9g20.bin at address 0x200000
-E- Error during external RAM initialization.
-E- External RAM access is required to run applets.
-E- Connection abort
```

9.3 Product Specification

For the Eddy series contents are as follows.

9.3.1 Eddy-CPU v2.5 Specifications

	Division	Specification
		Eddy-CPU v2.5
Hardware	CPU	AT91SAM9G20 (400 MHz)
	Memory	8MB Data Flash, 32 MB SDRAM
	External I/F	19 Bit / 16 Bit Data Bus
	Ethernet I/F	10/100 Base-T Auto MDI/MDIX
	UARTs	4 Port, Support up to 921.6 Kbps (1 : Full Signal, 2,3,4, : RxD, TxD, RTS, CTS only)
	USB 2.0 FS	2 Host /1 Device Port, 2.0 FS (12Mbps)
	ADC	4-Channel 10 Bit ADC
	TWI(I2C)	Master, Multi-Master and Slave Mode
	SPI	8- to 16-bit Programmable Data Length Four External Peripheral Chip Selects
	GPIO	Max. 56 Programmable I/O Pins
	Power Input	3.3 V (200 mA Max)
	Dimensions	25 x 48.5 x 6.2 mm
	Weight	8.3 g
Network	Protocol	TCP, UDP, Telnet, ICMP, DHCP, TFTP, HTTP, SNMP 1&2, SSH, SSL
	Ethernet	10/100Mbps MAC / PHY
	Network Connection	Static IP, DHCP
Software	O/S	Linux Kernel 2.6.21
	Mgt Tools	SNMP, Web, PortView
	Uploads	TFTP, FTP, Web

	Division	Specification
		Eddy-CPU v2.5
	Dev Tools	LemonIDE & SDK
Environmental	Operating Temp	-40 ~ 85 ℃
	Storage Temp	-60 ~ 150 ℃
	Humidity	5 ~ 95% Non-Condensing
Approvals	CE Class A, FCC Class A, RoHS compliant	CE FC RoHS RoHS Compliant

9.3.2 Eddy-DK v2.1 Specifications

Division	Specification
NAND Flash	256MB, 8bit I/F
SD Card Connector	Push Type, Up to 16 GB MMC / SD Card / MC supported
USB Connector	1 x Device 2 x HOST, Dual-Port
LCD Module	128 x 64 Dots Matrix Structure
KEY	4 x 4 Matrix
Battery Holder	3V Lithium Battery, 235 mAh
LED	Power, Ready, 20 Programmable IO, Console & Serial TxD, RxD
I2C Interface	16bit I2C BUS GPIO
SPI Interface	2 Kbit EEPROM
MCI Interface	SD Card, MMC Socket
ADC Interface	Temp / Light Sensor
Digital I/O	8 Port Input, 8 Port Output

Division	Specification
Switch	- Serial or GPIO Select - RS422/485 Select - DIO : Common VCC or GND Select - Programming
Jumper Switch	Boot Mode Select, JTAG Select
Serial Port	2 x RS232 DB9 Male 2 x RS422/485 Terminal Block (RS422 & RS485 Selected by S/W)
Console Port	DB9 Male
LAN Port	2 x RJ45
ICE Port	Used for Flash Programming
Reset Button	Factory Default & Warm Boot
Input Power	9-48VDC
Dimensions	240 x 180 mm

9.3.3 Eddy-S4M v2.5 Specifications

	Division	Specification
		Eddy S4M v2.5
Hardware	CPU	AT91SAM9G20 (400MHz)
	Memory	AT45DB642D, 8MB Data Flash IS42S16160B, 32 MB SDRAM
	Ethernet MC/PHY	10/100 Base-T MAC KSZ8041NLi PHYceiver Auto MDI/MDIX
	Serials	Port 0,1 : RS232 (DB9 male) Port 0 : Full Signal Port 1 : TxD, RxD, RTS, CTS only Port 2,3 : COMBO (Terminal Block 5pin) * COMBO : RS422/RS485 is S/W selectable

	Division	Specification
		Eddy S4M v2.5
	USB 2.0 FS	3 Host /1 Device Port, 2.0 FS (12Mbps) Expanded port by using GL850A USB Hub chip.
	RTC	Real Time Clock, RTC DS1340U-33+ Connect to I2C I/F
	Battery Holder	CR1220(38mAh) 3V Lithium Battery
	ADC	4-Channel 10 Bit ADC
	TWI(I2C)	Master, Multi-Master and Slave Mode
	SPI	8 to 16-bit Programmable Data Length Four External Peripheral Chip Selects
	MCI	Support SD Spec V2.0 [SDHC], MMC Spec V4.2 Applied USB to SD Controller, 16GB, 12Mbits/s
	GPIO	Max. 34 Programmable I/O Pins
	LED	Ready LED
Software	Protocol	TCP, UDP, Telnet, ICMP, DHCP, TFTP, HTTP, SNMP1&2, SSH, SSL
	Network Connection	Static IP, DHCP
	O/S	Linux Kernel 2.6.21
	Mgt Tools	SNMP, Web, PortView
	Uploads	TFTP, FTP, Web
	Dev Tools	LemonIDE & SDK
Physical characteristics	Power Input	3.3 V (Max. 200mA)
	Dimensions	59.75 x 61.80 x 4 mm
	Weight	15 g
Environment	Operating Temp	-40 ~ 85℃
	Storage Temp	-66 ~ 150℃
	Humidity	5 ~ 95% Non-Condensing

	Division	Specification
		Eddy S4M v2.5
CE Class A, FCC Class A, RoHS compliant	CE FC RoHS RoHS Compliant	

9.3.4 Eddy-S4M-DK v2.1 Specifications

Division	Specification
Serial Port	2 x RS232 DB9 Male 2 x RS422/485 5pin Terminal Block (Auto toggle selectable with S/W)
SD Card Connector	Push Type, up to 16 GB MMC / SD Card / MC supported
MCI Interface	SD Card, MMC Socket
ADC Interface	Light Sensor
USB Connector	1 x Device, 2 x HOST, Dual-Port
LAN Port	RJ45 with transformer
Console Port	DB9 Male
Switch	Power On/Off switch Serial RS422/485 Termination resistor setting switch Input GPIO test switch (Off : Low, ON : High)
LED	RDY, Power, 34 Programmable IO, Console & Serial TxD, RxD LED
JTAG Port	Used for downloading code and single-stepping through programs
Reset Button	Factory Default & Warm Boot (If pressed for 5 seconds or more, operates as factory default)
JIG connection socket	2 2 x 23pin socket, connector to connect with JIG board and test
Expansion Header	2 x 22pin Header, Connector to test GPIO in Eddy-S4M
Input Power	5 VDC
Dimensions	160 x 120 mm

9.3.5 Eddy-S4M-JIG v2.1 Specifications

Division	Specification
USB Connector	USB HOST
LAN Port	RJ45

Division	**Specification**
Reset Button	Factory Default & Warm Boot
Expansion Header	To provide all the features from S4M that can be connected with external devices.
Input Power	5 VDC
Dimensions	70 x 105 mm

9.3.6 Eddy-WiFi v3.0 Specifications

Classification	Specification
Standard	802.11b, 802.11g, 802.11n
Modulation	802.11b:CCK, DQPSK, DBPSK 802.11g:64 QAM, 16 QAM, QPSK, BPSK 802.11n:BPSK, QPSK, 16-QAM, 64-QAM
Frequency Band	ISM band 2.4GHz ~ 2.4884GHz
Output Power	802.11b:16 dBm (11Mbps) 802.11g:14 dBm (54Mbps) 802.11n:14 dBm (20MHz BW,MCS7) 13 dBm (40MHz BW,MCS7)
RX sensitivity	802.11b:-84dBm@11MHz 802.11g:-73dbm@54MHz 802.11n:-71dBm(MCS 7_HT20) -68dBm(MCS 15_HT20) -68dBm(MCS 7_HT40) -65dBm(MCS 15_HT40)
Security	WPA, WPA-PSK, WPA2, WPA2-PSK , WEP 64bit & 128bit , IEEE 802.11x, IEEE 802.11i
Working distance	60 - 120m, depending on surrounding environment
Data Rate	802.11b: 11, 5.5, 2, 1 802.11g: 54, 48, 36, 24, 18, 12, 9, 6 802.11n: 20 MHz BW: 130, 1117, 104, 78, 65, 58.5, 52, 39, 26, 19.5, 13, 6.5 40 MHz BW: 270, 243, 216, 162, 150, 135, 121.5, 108, 81, 54, 40.5,

	27, 13.5 (unit: Mbps)
Antenna	ANT 2.4Ghz 2DB, 1 x U.FL
Dimension	28.2 x 45.4 x 9.6 mm
Operating Temp	-10 ~ 70℃
Operating Voltages	3.3V ± 5% I/O supply voltage
Weight	10g
Approvals	KC, RoHS Compliant

9.3.7 Eddy-BT v2.1 Specifications

Division	Specification
Interface	Bluetooth v2.0+ EDR Class 1
Profile	SPP (Serial Port Profile)
Max, TX Power	+18dBm
RX sensitivity	-88dBm
Power	Supply voltage: 3.3V DC Supply current::10mA – 60mA
Operating Temp	Operating temperature: -30 ~ 80 ºC
Storage Temp	Storage temperature: -40 ~ 85 ºC
Humidity	Humidity : 90% (Non-condensing)
Working distance	Stub Antenna (+1dBi) - Stub Antenna (+1dBi) 100 meters Stub Antenna (+1dBi) - Dipole Antenna (+3dBi) 150 meters Dipole Antenna (+3dBi) - Dipole Antenna (+3dBi) 200 meters Dipole Antenna (+3dBi) - Dipole Antenna (+5dBi) 300 meters Dipole Antenna (+3dBi) - Patch Antenna (+9dBi) 500 meters Dipole Antenna (+5dBi) - Dipole Antenna (+5dBi) 400 meters Dipole Antenna (+5dBi) - Patch Antenna (+9dBi) 600 meters Patch Antenna (+9dBi) - Patch Antenna (+9dBi) 1,000 meters
Approvals	CE Class A, FCC Class A, RoHS Compliant

9.3.8 Eddy-CPU/mp v2.5 / v2.5 32bit Specifications

	Division	Specification
Hardware	CPU	AT91SAM9G20 (400 MHz)
	Memory	8MB Data Flash, 32 MB SDRAM, 64MB SDRAM
	External I/F	16 Bit / 32 Bit Data Bus
	Ethernet I/F	10/100 Base-T Auto MDI/MDIX

	Division	Specification
	UARTs	4 Port, Support up to 921.6 Kbps (1 : Full Signal, 2,3,4, : RxD, TxD, RTS, CTS only)
	USB 2.0 FS	2 Host /1 Device Port, 2.0 FS (12Mbps)
	ADC	4-Channel 10 Bit ADC
	TWI(I2C)	Master, Multi-Master and Slave Mode
	SPI	8- to 16-bit Programmable Data Length Four External Peripheral Chip Selects
	GPIO	Max. 56 Programmable I/O Pins
	Power Input	3.3 V (200 mA Max)
	Dimensions	59.75 x 44.6 X 1.0 mm
	Weight	8.3 g
Network	Protocol	TCP, UDP, Telnet, ICMP, DHCP, TFTP, HTTP, SNMP 1&2, SSH, SSL
	Ethernet	10/100Mbps MAC / PHY
	Network Connection	Static IP, DHCP
Software	O/S	Linux Kernel 2.6.21
	Mgt Tools	SNMP, Web, PortView
	Uploads	TFTP, FTP, Web
	Dev Tools	LemonIDE & SDK
Environmental	Operating Temp	-40 ~ 85 ℃
	Storage Temp	-60 ~ 150 ℃
	Humidity	5 ~ 95% Non-Condensing
Approvals	CE Class A, FCC Class A, RoHS compliant	CE RoHS RoHS Compliant

9.4 Ordering Information

Ordering information for Eddy product line is as follows.

Product	Version	Description
Eddy-CPU	**2.1**	Embedded CPU Module
Eddy-CPU	**2.5**	Embedded CPU Module
Eddy-CPU	**2.5B**	Embedded CPU Module **(64MB SDRAM)**
Eddy-DK	**2.1**	Eddy Development Kit with Eddy-CPU v2.1
Eddy-DK	**2.5**	Eddy Development Kit with Eddy-CPU v2.5
Eddy-DK	**2.5B**	Eddy Development Kit with Eddy-CPU v2.5 **(64MB SDRAM)**
Eddy-S4M	**2.1**	Embedded CPU Module (Mini PCI Type)
Eddy-S4M	**2.5**	Embedded CPU Module (Mini PCI Type)
Eddy-S4M-DK	**2.1**	Eddy-S4M v2.1 Development Kit
Eddy-S4M-JIG	**2.1**	Eddy-S4M v2.1 JIG Board
Eddy-WiFi	**3.0**	802.11 b/g/n Wi-Fi Module
Eddy-BT	**2.1**	Bluetooth 2.0 Module
Eddy-CPU/mp	**2.5**	Embedded CPU Module **(32MB SDRAM)**
Eddy-CPU/mp 32bit	**2.5B**	Embedded CPU Module **(64MB SDRAM)**
Eddy-CPU/mp-JIG	**2.5**	Eddy-CPU/mp v2.5 JIG Board