Jennic
TECHNOLOGY FOR A CHANGING WORLD

Board API

Reference Manual

JN-RM-2003
Revision 1.4
8-Jan-2007

Disclaimer

The contents of this document are subject to change without notice. Customers are advised to consult with JENNIC commercial representatives before ordering.

The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended for incorporation in devices for actual use. Also, JENNIC is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.

No license is granted by its implication or otherwise under any patent or patent rights of JENNIC Ltd

"Typical" parameters which are provided in this document may vary in different applications and performance may vary over time. All operating parameters must be validated for each customer application by the customer's own technical experts.

CAUTION:
Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, vehicle operating controls, medical devices for life support, etc.) are requested to consult with JENNIC representatives before such use. JENNIC customers using or selling products incorporating JENNIC IP for use in such applications do so at their own risk and agree to fully indemnify JENNIC for any damages resulting from such improper use or sale.

Contents

About this Manual

This manual provides a detailed reference for the Application Programming Interface (API) supplied with Jennic JN5121 and JN513x evaluation kits – the Board API.

The evaluation kits provide a low cost hardware platform for developing 802.15.4 wireless network applications (including ZigBee). The API allows rapid software development by providing a function library for interfacing to the evaluation board components.

Note: This manual was previously called the Evaluation Kit Library Reference Manual.

Organisation

This manual consists of two chapters:

- Chapter 1 Introduces the evaluation kit features.
- Chapter 2 describes the API function calls available for each feature.

Conventions

Code fragments or function prototypes are represented by `Courier` typeface. When referring to constants or functions defined in the code they are emboldened, **`like so`**

Acronyms and Abbreviations

API	Application Programming Interface
LED	Light Emitting Diode
LCD	Liquid Crystal Display
UART	Universal Asynchronous Receive Transmit

Revision History

Version	Date	Description
1.0	12-Sep-2005	First release
1.1	14-Nov-2005	Updated document style
1.2	10-Mar-2006	Removed references to specific evaluation kit
1.3	06-Oct-2006	Name of API changed from Evaluation Kit Library to Board API
1.4	08-Jan-2007	Updated for JN513x chip series

1 Introduction

1.1 Scope

This document describes the Application Programming Interface (API) for the Jennic JN5121 and JN513x evaluation kit boards – the Board API. Its functionality is as follows:

- LED control
- Key processing
- LCD panel
- Temperature sensor
- Humidity sensor
- Light sensor

Note: This API was previously known as the Evaluation Kit Library (as well as the Board API).

The Board API provides a thin layer above the registers used to control the evaluation kit board components, to encapsulate several register accesses into one function call and hence make it easier to use the peripherals without having to acquire detailed knowledge of their operation.

This document does not describe the API for features found on the JN5121/JN513x chip itself, such as the UARTs or DACs. The API for chip features is described in [1].

There are two evaluation kit boards, designated Controller and Endpoint boards.

- The Controller board has an LCD panel, four buttons and four LEDs
- The Endpoint has two buttons and two LEDs

In all other respects, the two boards have identical capabilities.

2 API Description

The API is described in terms of the functions used to access it, and they are grouped by peripheral namely the light sensor, humidity and temperature sensor, LCD panel, LEDs, and switches. The functions are defined in **`AlsDriver.h`**, **`HtsDriver.h`**, **`LcdDriver.h`**, **`LedControl.h`** and **`Button.h`** respectively

2.1 Ambient Light Sensor

The ambient light sensor provides an indication of the light level falling on the board. The functions are defined in **`AlsDriver.h`**.

2.1.1 vAlsReset

Declaration `PUBLIC void vAlsReset(void);`

Inputs None

Outputs None

Description Used to initialise the ambient light sensor. This should be called before any other accesses to the ambient light sensor are attempted.

2.1.2 vAlsStartReadChannel

Declaration `PUBLIC void vAlsStartReadChannel(uint8 u8Channel);`

Inputs `u8Channel` 0 for the channel that sees both visible and infra-red light
1 for the channel that sees infra-red light only

Outputs None

Description Used to initiate a read on one of the two channels available on the ambient light sensor. In the demo application only channel 0 is used, and after the first call to this function the device continually restarts conversions so there is no need to call it again.

2.1.3 u16AlsReadChannelResult

Declaration `PUBLIC uint16 u16AlsReadChannelResult(void);`

Inputs None

Outputs `uint16` Light level, in range 0 to 4015

Description Read the most recent light level reading. The value is approximately linear.

2.2 Temperature and Humidity Sensor

The temperature and humidity sensor provides an indication of the temperature and humidity levels in the demo board environment. The functions are defined in **`HtsDriver.h`**.

2.2.1 vHtsReset

Declaration `PUBLIC void vHtsReset (void);`

Inputs None

Outputs None

Description Used to initialise the combined humidity and temperature sensor. This should be called before any other accesses to the combined humidity and temperature sensor are attempted.

2.2.2 vHtsStartReadTemp

Declaration `PUBLIC void vHtsStartReadTemp(void);`

Inputs None

Outputs None

Description Used to initialise a temperature read. This should be called before each attempt to read the temperature.

2.2.3 u16HtsReadTempResult

Declaration `PUBLIC int16 i16HtsReadTempResult(void);`

Inputs None

Outputs `int16` Temperature in degrees C, range is -40 to 124

Description Used to read the most recent temperature reading from the sensor. This call blocks until the result is available.

2.2.4 vHtsStartReadHumidity

Declaration `PUBLIC void vHtsStartReadHumidity(void);`

Inputs None

Outputs None

Description Used to initialise a humidity read. This should be called before each attempt to read the humidity.

2.2.5 u16HtsReadHumidityResult

Declaration `PUBLIC uint16 u16HtsReadHumidityResult(void);`

Inputs None

Outputs `uint16` Relative humidity in %, range is 0 to 160

Description Used to read the most recent humidity reading from the sensor. This call blocks until the result is available.

2.3 LED Control

The LED control module provides a method of driving the LEDs on the demo board. The functions are defined in **`LedControl.h`**.

2.3.1 vLedInitRfd

Declaration `PUBLIC void vLedInitRfd(void);`

Inputs None

Outputs None

Description Used to initialise the 2 LEDs on the RFD (endpoint) board.

2.3.2 vLedInitFfd

Declaration `PUBLIC void vLedInitFfd(void);`

Inputs None

Outputs None

Description Used to initialise the 4 LEDs on the FFD (coordinator) board.

2.3.3 vLedControl

Declaration `PUBLIC void vLedControl(uint8 u8Led, bool_t bOn);`

Inputs `u8Led` Which LED to control, 0-1 for Endpoints and 0-3 for Controller

`bOn` TRUE to turn LED on, FALSE to turn it off

Outputs None

Description Used to control an individual LED.

2.4 Button Input

The button input module provides a way of determining the buttons that are pressed on the demo board. The functions are defined in **`Button.h`**.

2.4.1 vButtonInitRfd

Declaration `PUBLIC void vButtonInitRfd(void);`

Inputs None

Outputs None

Description Used to initialise the 2 buttons on the RFD (endpoint) board. Should be called before any accesses to the other button functions are attempted.

2.4.2 vButtonInitFfd

Declaration `PUBLIC void vButtonInitFfd(void);`

Inputs None

Outputs None

Description Used to initialise the 4 buttons on the FFD (coordinator) board. Should be called before any accesses to the other button functions are attempted.

2.4.3 u8ButtonReadRfd

Declaration `PUBLIC uint8 u8ButtonReadRfd(void);`

Inputs None

Outputs `uint8` Result logical and with BUTTON_0_MASK is non-zero if button 0 is pressed
Result logical and with BUTTON_1_MASK is non-zero if button 1 is pressed

Description Used to read the button states on the RFD (endpoint) board. Note that there is no de-bounce circuit or algorithm employed. It is possible and legitimate for several buttons to be pressed at once.

2.4.4 u8ButtonReadFfd

Declaration `PUBLIC uint8 u8ButtonReadFfd(void);`

Inputs None

Outputs `uint8` Result logical and with BUTTON_0_MASK is non-zero if button 0 is pressed
Result logical and with BUTTON_1_MASK is non-zero if button 1 is pressed
Result logical and with BUTTON_2_MASK is non-zero if button 2 is pressed

Result logical and with BUTTON_3_MASK is non-zero if button 3 is pressed

Description Used to read the button states on the FFD (coordinator) board. Note that there is no de-bounce circuit or algorithm employed. It is possible and legitimate for several buttons to be pressed at once.

2.5 LCD Panel

2.5.1 Hardware Features

The LCD panel on the demo board can be used to display both text and graphics. The functions are defined in **`LcdDriver.h`**.

The LCD panel has a resolution of 64 rows and 128 columns, but is internally arranged so that one byte contains the pixel information for a single column of 8 rows. As a result, the driver is considerably simplified by only allowing positioning of text or graphics on 8 row boundaries. Herein, a block of 8 rows is referred to as a 'character row', with the LCD panel containing 8 character rows. There is no such limitation on the columns.

2.5.1.1 Shadow memory

The LCD driver makes use of a shadow of the LCD contents. This allows a screen of information to be built in the shadow memory, and a command can then update the LCD panel with the contents of the shadow in one go, improving performance and minimising any danger of seeing a partially changed screen.

2.5.1.2 LCD font

The font available for text is proportional, with most characters being 5 pixels wide, though several are narrower and there are also some special characters used for the demo application which are 7 pixels wide. All characters are 8 pixels high. When printing text, there is a blank column before the first character, between each character and after the last character.

The font is mapped approximately to the ASCII character map, although some characters are moved to simplify the font processing. The map is as follows:

ASCII code	ASCII character	LCD font character
37	'%'	Percent symbol
38-44	'&'-','	Full dark moon – full light moon symbols
48-57	'0'-'9'	'0'-'9'
65-90	'A'-'Z'	'A'-'Z'
91	'['	Degrees symbol
92	'\'	Plus symbol
93	']'	Minus symbol
94	'^'	Space
97-122	'a'-'z'	'a'-'z'

Other characters are displayed as space.

2.5.1.3 LCD bitmaps

Bitmaps are defined to be simple to render and as such map to the way that the LCD panel maps pixels. A bitmap can be any number of columns but must always be a multiple of eight bits high. Each bitmap is treated as a 'C' structure consisting of the width of the bitmap in pixels, the height in character rows and a pointer to an array of uint8 (bytes) containing the pixel data.

```
typedef struct
{
    uint8 *pu8Bitmap;
    uint8 u8Width;
    uint8 u8Height;
} tsBitmap;
```

Consider the bitmap shown here, y columns wide and 16 rows high. Each pixel is represented by a two letter name, e.g. aa, ab, etc.

		column					
		0	**1**	**2**		**y-2**	**y-1**
row	**0**	aa	ab	ac	..	aw	ax
	1	ba	bb	bc	..	bw	bx
	2	ca	cb	cc	..	cw	cx
		..	..	..	..	..	..
	7	ha	hb	hc	..	hw	hx
	8	ia	ib	ic	..	iw	ix
	9	ja	jb	jc	..	jw	jx
		..	..	..	..	..	..
	15	pa	pb	pc	..	pw	px

The first element in the array of pixel data contains the first column of the top character row of pixels, i.e.

pu8Bitmap[0] = (MSB) ha ga fa ea da ca ba aa (LSB)

The second element in the array of pixel data contains the second column of the top character row of pixels, i.e.

pu8Bitmap[1] = (MSB) hb gb fb eb db cb bb ab (LSB)

And so on to the end of the first character row, so the final column of the first row is in element y-1 of the array, i.e.

pu8Bitmap[y-1] = (MSB) hx gx fx ex dx cx bx ax (LSB)

The first column of the second character row of pixels is the next element in the array, i.e.

pu8Bitmap[y] = (MSB) pa oa na ma la ka ja ia (LSB)

This row continues until the final column, which will be in element 2y-1, i.e.

pu8Bitmap[2y-1] = (MSB) px ox nx mx lx kx jx ix (LSB)

If there were further rows, they would repeat in the same manner.

2.5.2 vLcdResetDefault

Declaration `PUBLIC void vLcdResetDefault(void);`

Inputs None

Outputs None

Description Used to initialise the LCD panel using default settings for bias and gain, which should give a good level of contrast. The LCD screen is cleared.

2.5.3 vLcdReset

Declaration `PUBLIC void vLcdReset(uint8 u8Bias, uint8 u8Gain);`

Inputs

`u8Bias`	Bias value to use, 0 to 3 are valid
`u8Gain`	Gain value to use, 0 to 3 are valid

Outputs None

Description Used to initialise the LCD panel using specific settings for bias and gain. The LCD screen is cleared.

2.5.4 vLcdStop

Declaration `PUBLIC void vLcdStop(void);`

Inputs None

Outputs None

Description Used to turn off the LCD. This is normally only used before shutting down the demo board to allow the LCD to discharge itself properly.

2.5.5 vLcdClear

Declaration `PUBLIC void vLcdClear(void);`

Inputs None

Outputs None

Description Clears the shadow memory of any text or graphics. Does not update the LCD itself. Use **`vLcdResfreshAll()`** or **`vLcdRefreshArea()`** to update the shadow memory to the LCD.

2.5.6 vLcdRefreshAll

Declaration `PUBLIC void vLcdRefreshAll(void);`

Inputs None

Outputs None

Description Copies the contents of the shadow memory to the LCD panel. This takes approximately 4.5ms.

2.5.7 vLcdRefreshArea

Declaration
```
PUBLIC void vLcdRefreshArea(uint8 u8LeftColumn,
                            uint8 u8TopRow,
                            uint8 u8Width,
                            uint8 u8Height);
```

Inputs

`u8LeftColumn`	Left-most column to update (0-127)
`u8TopRow`	Top row to update (0-7)
`u8Width`	Number of columns to update (1-128)
`u8Height`	Number of rows to update (1-8)

Outputs None

Description Copies the contents of the specified rectangle of the shadow memory to the appropriate part of the LCD panel without disturbing the rest of it.

2.5.8 vLcdWriteText

Declaration
```
PUBLIC void vLcdWriteText(char *pcString,
                          uint8 u8Row,
                          uint8 u8Column);
```

Inputs

`pcString`	Null-terminated text string to display
`u8Row`	Row on which to display text (0-7)
`u8Column`	Column on which to start displaying text (0-127)

Outputs None

Description Prints a text screen to the shadow memory. The text is left-justified and starts at the row and column specified. No attempt is made to prevent the text from spilling past the end of the current row, and if this occurs it will wrap around to the next row. Use **`vLcdResfreshAll()`** or **`vLcdRefreshArea()`** to update the shadow memory to the LCD.

2.5.9 vLcdWriteTextRightJustified

Declaration
```
PUBLIC void vLcdWriteTextRightJustified(char *pcString,
                                        uint8 u8Row,
                                        uint8 u8EndColumn);
```

Inputs

`pcString`	Null-terminated text string to display
`u8Row`	Row on which to display text (0-7)
`u8EndColumn`	Column on which to finish displaying text (0-127)

Outputs None

Description Prints a text screen to the shadow memory. The text is right-justified and finishes at the row and column specified. No attempt is made to prevent the text from spilling past the start of the current row, and if this occurs it will wrap around to the previous row. Use **`vLcdResfreshAll()`** or **`vLcdRefreshArea()`** to update the shadow memory to the LCD.

2.5.10 vLcdWriteInvertedText

Declaration
```
PUBLIC void vLcdWriteInvertedText(char *pcString,
                                  uint8 u8Row,
                                  uint8 u8Column);
```

Inputs

`pcString`	Null-terminated text string to display
`u8Row`	Row on which to display text (0-7)
`u8Column`	Column on which to start displaying text (0-127)

Outputs None

Description This functions as for **`vLcdWriteText()`** except the text is inverted. This can be used for highlighting. Use **`vLcdResfreshAll()`** or **`vLcdRefreshArea()`** to update the shadow memory to the LCD.

2.5.11 vLcdWriteBitmap

Declaration
```
PUBLIC void vLcdWriteBitmap(tsBitmap *psBitmap,
                            uint8 u8LeftColumn,
                            uint8 u8TopRow);
```

Inputs

`psBitmap`	Pointer to structure containing bitmap information
`u8LeftColumn`	Leftmost column of bitmap (0-127)
`u8TopRow`	Top character row of bitmap (0-7)

Outputs None

Description Puts the bitmap into the shadow memory at the location specified. If the bitmap goes past the edge of the display area it is truncated. Use **`vLcdResfreshAll()`** or **`vLcdRefreshArea()`** to update the shadow memory to the LCD.

2.5.12 vLcdPlotPoint

Declaration
```
PUBLIC void vLcdPlotPoint(uint8 u8X, uint8 u8Y)
```

Inputs

`u8X`	X coordinate of point to be plotted (0-127)
`u8Y`	Y-Coordinate of point to be plotted (0-63)

Outputs None

Description Plots a singe pixel in the shadow memory at coordinates (x,y). If either x or y is out of range no pixel is plotted. Use **`vLcdResfreshAll()`** to update the shadow memory to the LCD.

2.5.13 vLcdGetPixel

Declaration `PUBLIC bool_t vLcdPlotPoint(uint8 u8X, uint8 u8Y)`

Inputs	`u8X`	X coordinate of point to be plotted (0-127)
	`u8Y`	Y-Coordinate of point to be plotted (0-63)
Outputs	Bool_t	TRUE if pixel is set, FALSE if clear

Description Returns the status of pixel in the shadow memory at coordinates (x,y). If either x or y is out of range returns false.

2.5.14 vLcdDrawLine

Declaration
```
PUBLIC void vLcdDrawLine(uint8 u8x1, uint8 u8y1,
                         uint8 u8x2, uint8 u8y2)
```

Inputs	`u8x1`	Start X coordinate of line (0-127)
	`u8y1`	Start Y-Coordinate of line (0-63)
	`u8x2`	End X coordinate of point (0-127)
	`u8y2`	End Y-Coordinate of line (0-63)
Outputs	None	

Description Draws a singe pixel line from (x1, y1) to (x2, y2) in the shadow memory. The line is clipped if any points lie outside the screen area. Use **`vLcdResfreshAll()`** to update the shadow memory to the LCD.

2.5.15 vLcdDrawCircle

Declaration `PUBLIC void vLcdDrawCircle(int Xc,int Yc,int Radius)`

Inputs	`i32Xc`	Centre X coordinate of Circle (0-127)
	`i32Yc`	Centre Y-Coordinate of line (0-63)
	`i32Radius`	Radius of circle
Outputs	None	

Description Draws a circle centred on (Xc, Yc) with given radius in the shadow memory. The circle is clipped if any points lie outside the screen area. Use **`vLcdResfreshAll()`** to update the shadow memory to the LCD.

2.5.16 vLcdFloodFill

Declaration `PUBLIC void vLcdFloodFill(int i32x, int i32y)`

Inputs

`i32x`	Start X coordinate of Fill (0-127)
`i32Y`	Start Y-Coordinate of Fill (0-63)

Outputs None

Description Flood fills a region within the shadow memory. Starting coordinates must be inside region and not on boundary for algorithm to work. In addition, region to be filled must have continuous bounder or fill will 'leak' and corrupt the display. Use **`vLcdResfreshAll()`** to update the shadow memory to the LCD.

References

[1] Jennic Integrated Peripherals API Reference Manual (JN-RM-2001)

Notes

Jennic is a fabless semiconductor company leading the wireless connectivity revolution into new applications. Jennic combines expertise in systems and software with world class RF and digital chip design to provide low cost, highly integrated silicon solutions for its customers and partners. Headquartered in Sheffield, UK, Jennic is privately held and has a proven track record of successful silicon chip development.

Jennic

TECHNOLOGY FOR A CHANGING WORLD

Jennic Ltd.
Furnival Street, Sheffield, S1 4QT, UK
Tel: +44 (0) 114 281 2655 Fax: +44 (0) 114 281 2951
Email: info@jennic.com Web: www.jennic.com

