SmartKit v0.01

Documentation



SmartKit v0.01

Raspberry Pi

ESP8266

Arduino

Internet Of Things

Android

2019

Vilasrao Khade & Ayush Khade

Cavy Agrotronics

01/01/2019

**INTRODUCTION**

**SmartKit v0.01 :**

Smartkit v0.01 manufactured and marketed by **Cavy Agrotronics** is programmable interfacing device with UART communication protocol ( TX and RX with specified baud rate).It work as INPUT/OUTPUT device with programmable 4X4 matrix **alphanumeric** keypad and 16X2 LCD display plus programmable nine **digital**  and five **analog** GPIO PINS which can be used for activating relay ,reading sensor ,driving servo etc .(useful in application with Arduino,Raspberry,esp8266 and IOT projects as a IO console).

It is very useful for electronic hobbyist , enthusiast ,programmers ,engineering student, teachers and IOT fans .It is language independent ,means no matters what programming language the **SmartKit v0.01** user is using in his project . User can interface SmartKit v0.01 with simple **Send-Response protocol**.

**Features:**

1)**Changeable Keypad layout :** SmartKit v0.01 have an uniquefeature of changing 4x4 matrix Keypad’s layout without changing a single wire and even writing a single line of code .After Keypads 16 KEYS defined ,this 4x4 matrix keypad covers all printable ASCII characters (Dec values 32 to 126).

**2) Baud rate setting** : The Baud rateof SmartKit v0.01Can be changed anytime for interfacing with PC/MCU or any other devices as per requirement of the application.

**3)Easily switchable two different Modes of operation** :

**Externally active:**

SmartKit v0.01 Externally active mode is default mode which is set for receiving command through serial communication. If the command is send to SmartKit through serial. The SmartKit will execute command and push response message to serial. In an Externally active mode the only (‘\*’) Key of Keypad of SmartKit v0.01 is active rest becomes active only after sending command <GetString()> through serial after which keypad is used for building string to send (by pressing ENTER Key)in response of the <GetString()> command.

**Internally active:**

You can switch from **Externally active mode** to **Internally active mode** only by holding(pressing for one second )  **‘\*’ Key** of Keypad (‘\*’ KEY is active in **Externally active mode** as stated earlier).In this Internally active mode you can send string continuously by typing string using matrix keypad and sending to serial with line feed (‘\n’ )by pressing **enter KEY.**

In this mode SmartKit accept the command typed from its own keypad i.e. if the string typed is a command for ex. #PinMode(0,OUTPUT) (note : String starting with **‘#’**) and Entered. SmartKit will identify the string, as a command to SmartKit and will execute this command by setting its digital pin D0 as OUTPUT and sending response string to serial . (The term response string in detail is discussed in next article **Getting started with SmartKit**).

Only pinMode(Pin,MODE),digitalWrite(Pin,value),analogRead(Pin),analogRead(Pin), AnalogWrite(Pin) and SetBaud(value) are valid commands for Internally active mode.

**4) Compatible with generic 4X4 matrix keypad:**

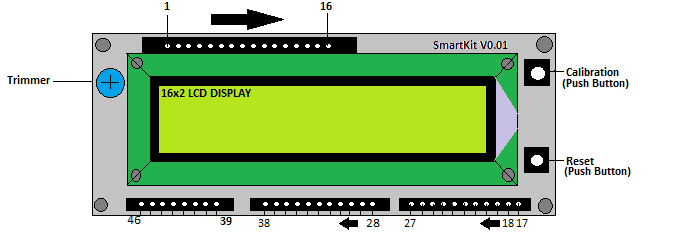
Keypad of SmartKit is analog keypad in which each combination of row column is designed to feed unique precise voltage to identify the KEY .This value of feed voltage may be different for Tactile type thin membrane keypad and tactile type push button keypad due the length of connectors, internal resistance and RF interference.

So using any type of keypad first time we need to configure to use it precisely. Configuring keypad mean defining keys for precise value.

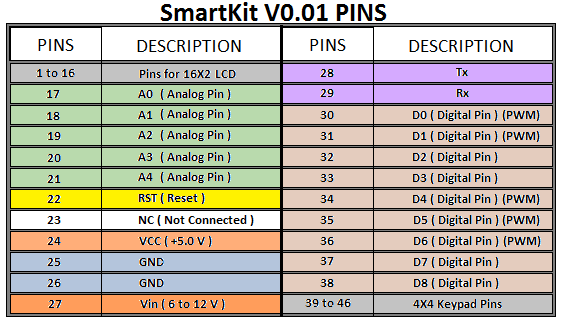
Ability to define KEYS of keypad is very useful and unique feature of changing 4x4 matrix Keypad’s layout without changing a single wire and even writing a single line of code as mention earlier .

Defining the KEYS of SmartKit v0.01 is very simple procedure and explained in details just after the Hardware Overview under head **Getting Started with SmartKit v0.01** .

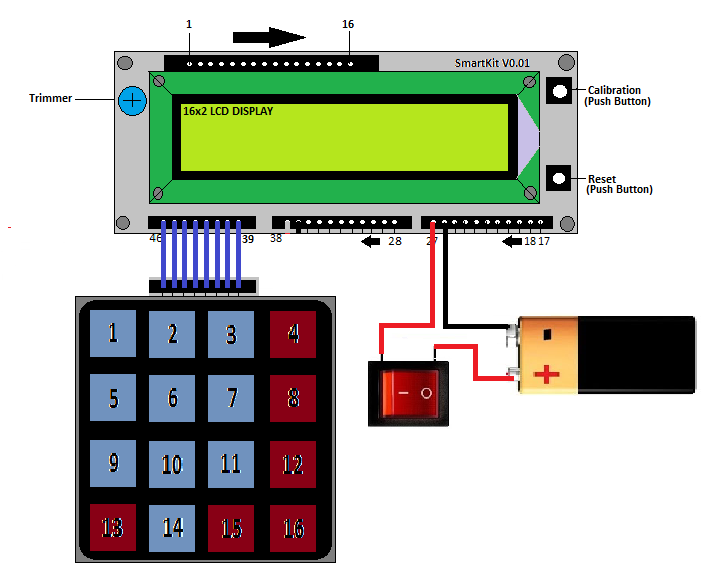
**HardWare Overview:**



.

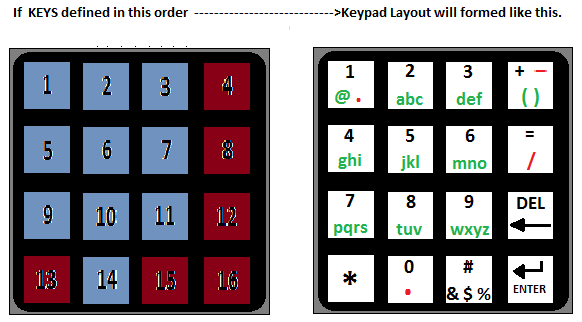
* **1 16 pin female header** which is suitable for easy mounting of 16X2 LCD **character display** (compatible with Hitachi 44780 driver).
* **2 8 pin male header** whichis suitable for making easy connection with 4X4 matrix keypad.
* **3 Reset Button(RST).**
* **4 Calibration Button(CAL).**
* **5 Trimmer potentiometer for adjusting brightness of LCD display.**
* **6 9 Digital (D0 to D8) and 5 analog(A0 to A4) GPIO PINS.**
* **7 Vcc (Regulated Supply voltage 5 V )**
* **8 Vin(6.5 to 12 volt battery supply)**
* **9 GND (Ground).**
* **10 RST (reset).**
* **11 TX ,RX (UART communication).** 

**Getting started with SmartKit**

****

Make connections as in diagram Switch ON the supply (DC 6 Volt to 9 Volt). SmartKit will respond with LCD light up .Adjust **Trimmer potentiometer** for adjusting brightness of LCD display to make the characters on LCD display visible. Now you will see the welcome message **“SmartKit V0.01”** on screen for with **Baud rate** defined**.**

For defining the keys of matrix Keypad .Hold **calibration button** and then resetSmartKit with **Reset button** (Both are **tactile** type push buttons) SmartKit v0.01 will display **“SmartKit v0.01”** and prompt “**Config KEYPAD “** message on second line of display .Now define keys in the order as shown in fig below to define KEYS hold each key until **“KEY(number) Defined!”** message appears on screen . Care must be taken for release of hold key quickly within a second after **“Key defined “** message displayed (Don’t worry if you fails to release Key after defined within the time ,SmartKit will take care of it and will display error and will ask you **“Try (next number) KEY again “** ,then again hold KEY which you have to define for that **KEY number** )this way you have to define 16 KEYS . After **“KEY(16) Defined!”**  message SmartKit will prompt for **“SET BAUD ? y/n”** . Press 1 to set baud rate using just configured keypad! if other than **KEY(1)**pressed baud rate will set to its default as 9600). After setting baud rate press **Enter KEY** SmartKit will display **version and Baud rate** it means it is configured and ready for application in your project.



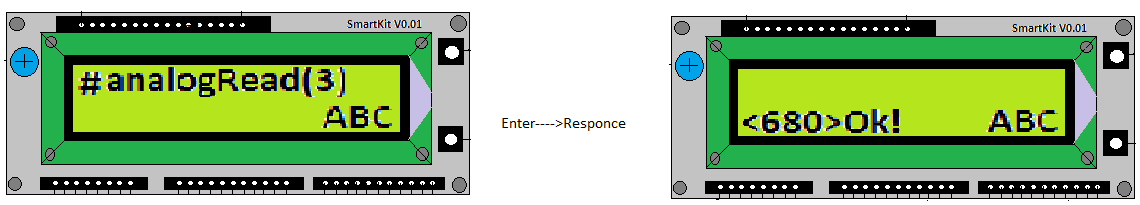
Details of defined KEYS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **MODE- HOLD ‘\*’ KEY(13) TO SWITCH BETWEEN MODES** | | | | - | - | - |
| **KEY N0.** | **Numeric**  **(123)** | **Small alphabets(abc)** | **Capital Alphabets(ABC)** | **Special Characters(#$%&)** | | | |
| **1** | 1 | @ and . (on hold) | @ and . (on hold) | . | | | |
| **2** | 2 | abc (toggle) | ABC (toggle) | **’** | | | |
| **3** | 3 | def (toggle) | DEF (toggle) | / | | | |
| **4** | + and **–** ( on hold) | ( ) (toggle) | ( ) (toggle) | :;<= (toggle) | | | |
| **5** | 4 | ghi (toggle) | GHI (toggle) | **,** | | | |
| **6** | 5 | jkl (toggle) | JKL (toggle) | **-** | | | |
| **7** | 6 | mno (toggle) | MNO (toggle) | [ \ ] (toggle) | | | |
| **8** | **=** and / ( on hold) | **,** | **,** | ! ” (toggle) | | | |
| **9** | 7 | pqrs (toggle) | PQRS (toggle) | ^ \_ ` (toggle) | | | |
| **10** | 8 | tuv (toggle) | TUV (toggle) | >? @ (toggle) | | | |
| **11** | 9 | wxyz (toggle) | WXYZ (toggle) | { | } ~ (toggle) | | | |
| **12** | Backspace | Backspace | Backspace | Backspace | | | |
| **13** | **\*** | **\*** | **\*** | **\*** | | | |
| **14** | 0 and . (on hold) | Space | Space | Space | | | |
| **15** | # $%& (toggle) | # $%& (toggle) | # $%& (toggle) | # $%& (toggle) | | | |
| **16** | Enter | Enter | Enter | Enter | | | |

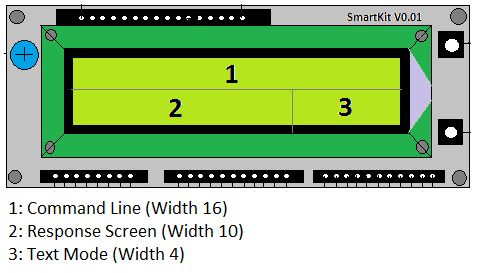
Now SmartKit v0.01 is in **Externaly active mode** ( default ) and is ready for receiving command through serial communication. Before going in details of interfacing methods in this mode we will look after its another mode of operation i.e. **Internally active mode**.

**Internally active mode**. Now to switch SmartKit to work in Internally active mode hold ‘\*’ key ! You will see the cursor on display starts blinking and keypads **numeric mode** (123) is displayed at the right bottom corner of 16x2 lcd . In this **Internally active mode** SmartKit’s working is unidirectional i.e. you can only send a string to Serial , Now type a string using alphanumeric keypad and to send it to serial press Enter KEY. (To verify what is received at serial monitor or hyper terminal we have not connected any terminal here we will see it in next article **How to interface serial** in the next example.)

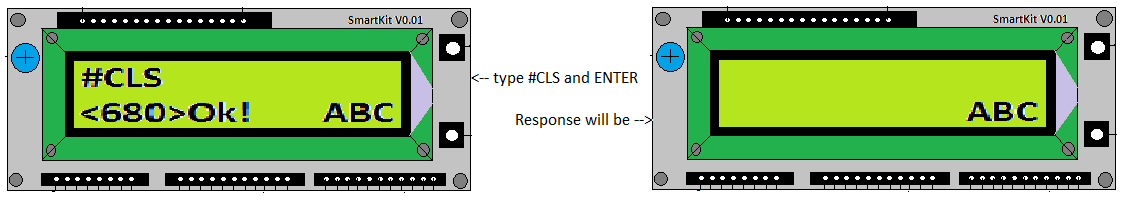
Only sending string ( built using keypad) to serial is not the usability of SmartKit ! Let us see how to use SmartKit in **Internally active mode.** Make connection as shown below. Now type #AnalogRead(3) and ENTER (by pressing Enter key). The response of SmartKit to the #AnalogRead(3) entered is shown in the picture below.



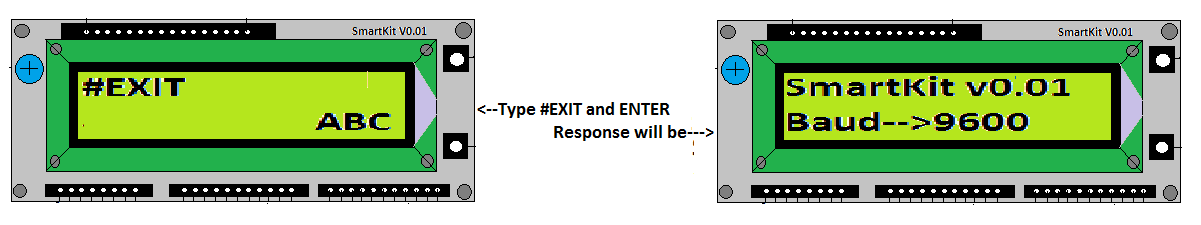
If the string entered is with starting letter **#** in Internally active mode the SmartKit process it with its internal command set and send response to the Response Screen of SmartKit. For **response screen** of SmartKit 10 char space is reserved on 2nd Row (left side) of LCD in Internally active mode. See layout of 16x2 LCD in the picture below for better understanding.



Now type # CLS and ENTER the response will be …

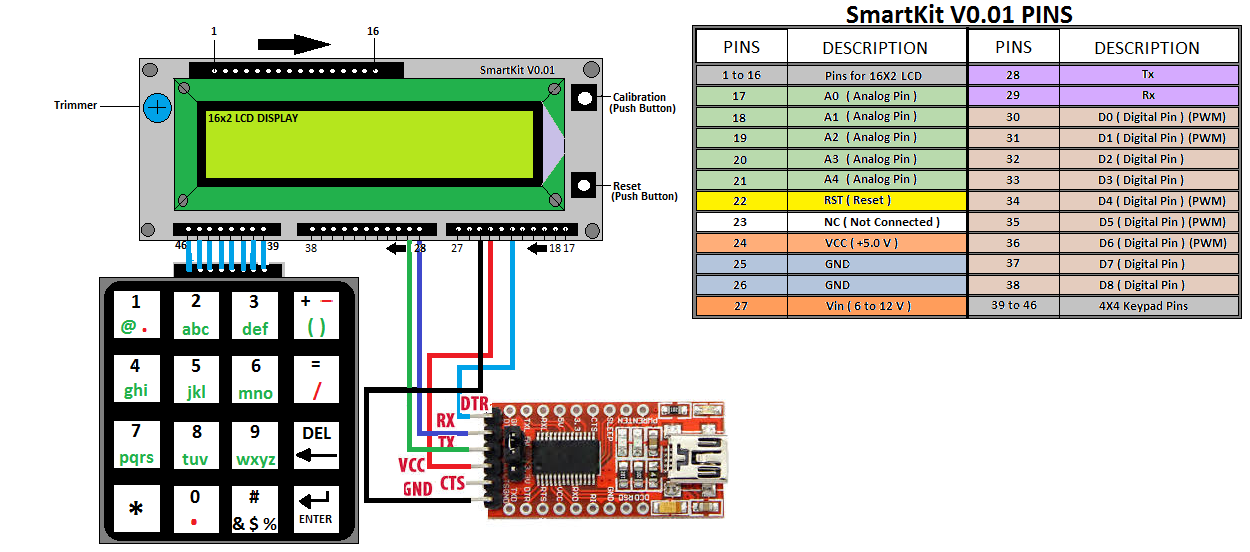


CLS is a command to clear response screen. (See the command document on page 8).

The **#CLS ,#BAUD , #EXIT** ,**#PinMode(Pin,Mode) , #DigitalWrite(Pin,Mode), #DigitalRead(Pin), #AnalogRead(Pin), #AnalogWrite(Pin,value), #SetBaud(value)** are commands for this Internally active mode. Now to switch SmartKIt from internally active mode to Externally active mode, simply type **#EXIT** andenter, response will be…

Now the SmartKit is in **externally active mode and** is ready for interfacing .Next article is in details about it.

# SmartKit Interface Basics

****

# (Here it is assumed that FTDI232RL driver is installed in your computer. If not then download CDM21228\_setup.exe from [www.driverscloud.com/en/services/GetInformationDriver/71116-58653/ftdi-chip-cdm21228-setupzip](http://www.driverscloud.com/en/services/GetInformationDriver/71116-58653/ftdi-chip-cdm21228-setupzip) and install it.)

Make connections as in diagram and insert USB cable of FT232RL breakout board in USB socket of computer . SmartKit will respond with LCD light up .Adjust **Trimmer potentiometer** for adjusting brightness of LCD display to make the characters on LCD display visible. Now you will see the welcome message **“SmartKit V0.01”** on screen for with **Baud rate** defined**.** (It is assumed that keypad of SmartKit is configured, if not then define the keys as per procedure given previously in article ‘Getting started with SmartKit’ ).

Now open **Arduino IDE** select proper port from Tool🡪port: COM7 .(this may be different allotted by your computer like COM4 ,COM5 in my case it was COM7 ).Now we will use Arduino serial monitor to understand interface basic better .( You can use any standard serial terminal program to connect to SmartKit v0.01, such as: Windows hyperterminal, Coolterm, puTTY, etc.)

**Go to menu Tool🡪Serial Monitor**

Look at the Serial Monitor window.

* - The small upper box is where you can type in command (hit or click "Send")
* - The larger area (Corner can be dragged to enlarge) is where characters sent From SmartKit will be displayed.- At the bottom are two pull downs:
* - One sets the "line ending" that will be sent to SmartKit when you or click Send (set ‘Both NL and CR’)
* - The other sets the Baud Rate for communications. (Set baud rate to match Baud rate of your SmartKit . If this does not match the Baud rate set for SmartKit , characters will be unreadable). Some sketches or other applications may use a different Baud Rate.

. **Now type <AnalogRead(A0)> in upper box and hit or click “Send”.** The SmartKit will push response message something like **<324>ok!** Where 324 is the value read from Analog pin(A0 ) of SmartKit. and if the command **<AnalogRead(10)>** is send to SmartKit .The SmartKit will push response message **>Error!.**Because SmartKit do not have analog pin(10) to read the values from.

The basic interface is to send SmartKit a line of Instruction within enclosed chevrons as in example **<AnalogRead(A0)>** then wait for the proper **response message** with open chevrons (**>**) followed by  **ok!**or **error!**( **“>ok!”** Or **“>Error!”** ) .This signals SmartKit has completed the parsing and executed the command. At times, SmartKit may not respond immediately. This happens when SmartKit is busy doing something else mostly happened with \***GetString()** .

The **response messages** are strings terminated by a return. These messages are "pushed" from SmartKit to the user in response to a query or instruction to let the user what happened.

The primary way to talk to SmartKit is performed by sending it a command string of characters ( Instruction within enclosed chevrons”<>” ), followed by a carriage return. SmartKit will then process the string, set it up for execution, and then reply back with a **response message**, also terminated by a return, to tell you how it went. These command strings include sending SmartKit: a Instruction code to execute, commands to configure SmartKit settings etc

The interface for Smartkit is fairly simple and straightforward. With SmartKit v0.01, steps have

been taken to try to make it even easier for new users to get started, and for GUI developers to write their own custom interfaces to SmartKit v0.01.

**Writing an Interface for SmartKit:**

**Streaming Protocol:** **Simple Send-Response** [Recommended]

The send-response streaming protocol is the most fool-proof and simplest method to stream a SmartKit command program to SmartKit. The host PC/ MCU interface simply sends a line of SmartKit command to SmartKit and waits for an ok or error: response message before sending the next line of SmartKit command. So, no matter if SmartKit needs to wait for room in the look-ahead planner buffer to finish parsing and executing the last line of SmartKit command or if the the host computer is busy doing something, this guarantees both to the host PC /MCU and SmartKit, the programmed SmartKit command has been sent and received properly. ( Note :All commands for SmartKit are **not Case Sensitive**! )

**See the Commands document to see what they are and how they work.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Commands | Example | Description | Response message | Remark |
| **<PinMode(Pin, Mode)>** | <PinMode(1,OUTPUT)> | Configures the D1 pin to behave as an output. | >Ok! |  |
| **<DigitalWrite(Pin,Value)>** | <DigitalWrite(1,HIGH)> | D1 pin voltage will be set to the 5V  HIGH(5V,LOW(0V) | >Ok! |  |
| **<AnalogWrite(Pin,value)>** | <AnalogWrite(1,55)> | Writes an analog value 55 to [PWM](http://arduino.cc/en/Tutorial/PWM)  Pin D1 | >Ok! | PWM Pins D0,D1,D4,D5,D6  Value range 0-255 |
| **<DigitalRead(Pin)>** | <DigitalRead(1)> | Reads the value from a specified digital pin D1, either HIGH or LOW. | <0>Ok! | 0 for LOW  1 for HIGH |
| **<AnalogRead(Pin)>** | <AnalogRead(A0)> | Reads the value from the specified analog pin A0 | <845>Ok! | 0 to 1024 for 0 to 5Volt |
| **<GetString(Prompt)>** | <GetString()>  //Prompt is optional | SmartKit accepts keys for building string until enter key pressed. | <String entered>Ok! | Smartkit will accept next command after string received. |
| **<WaitForKeyNum()>** | <WaitForKeyNum()> | Wait for the key press and returns the Key number of key pressed when released. | <3>Ok! | Means Key 3 was pressed |
| **<SetCursor(Col, Row)>** | <SetCursor(10,1)> | Set cursor of smartkit lcd to 10 th column and 2 nd row | >Ok! |  |
| **<LcdPrint(value)>** | 1)<LcdPrint(Yash)>  2)<LcdPrint(12.1413 ,2)>  3)<LcdPrint(“Ayush”)> | Will print🡪 Yash  Will print 🡪12.14  Will print🡪 “Ayush”  On SmartKits Display | >Ok! | Printing starts from the current Cursor position. |
| **<LcdClear()>** | <LcdClear()> | Clears smartkits lcd display | >Ok! | After Clear lcd cursor is at (0,0) |
| **<Cursor()>** | <Cursor()> | Underscore \_style LCD cursor will be displayed. | >Ok! |  |
| **<NoCursor()>** | <NoCursor()> | Hides the cursor | >Ok! |  |
| **<Blink()>** | <Blink()> | Cursor will blink until <noBlink()> command | >Ok! |  |
| **<NoBlink()>** | <noBlink()> | Opposite to< Blink()> | >Ok! |  |
| **<Help()>** | <Help()> | Send syntax of SmartKit commands to serial | 1: <pinMode(Pin, Mode)>  2: ………..  ...>Ok! |  |

**SmartKit Library for Arduino**

Library developed for interfacing with arduino is available at-<https://github.com/CavyAgrotronics/SmartKit> Now take a brief look on functions of SmartKit v 0.01 library.

**Functions for GPIO PINS control:**

1. **SetPort()**

**Description:** Defines the arduino pins and Baud rate for communication with SmartKit using software serial library.

**Syntax:** SetPort( rx, tx, rst, baud) where **rx** is the pin on which to receive serial data, **tx** is the pin on which to transmit serial data, **rst** is reset pin and long integer **baud.**

**Example:**SetPort(10,11,13,57600);

**Returns:**Nothing

**Example Code:**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ -------

#include<SmartKit.h>

String name;

SmartKit KP;//creates a variable of type SmartKit

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//Baud rate of SmartKit v0.01

void setup(){

KP.SetPort(rx,tx,rst,baud);//Defines the port for communication with SmartKit

}

void loop(){

//Do something here

}//end of loop

-------------------------------------------------------------- (End) -----------------------------------------------------------------

1. **PinMode():**

**Description:** Configures the specified pin of **SmartKit** to behave either as an input or output.

**Syntax:** PinMode(Pin,Mode)

Pin : the number of the SmartKit Pin whose mode you wish to set.

Mode:INPUT,OUTPUT or INPUT\_PULLUP

**Example:**PinMode(2,OUTPUT);

**Returns:**Nothing

1. **DigitalWrite()**

**Description:** Write HIGH or a LOW value to a digital pin of SmartKit.

If the pin has been configured as an OUTPUT with PinMode(),its voltage will be set to the corresponding value : 5V for HIGH ,0 V for LOW. If the pin is configured as an INPUT .It is recommended to set the pin PinMode () to INPUT\_PULLUP to enable internal pull-up resistor.

**Syntax:** DigitalWrite( Pin,Value ) ;

Pin: Is the pin number

Value : HIGH or LOW**.**

**Example:** DigitalWrite(2,HIGH);

**Returns:** Nothing

**Example Code:**

**/\* Ex:1**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

**D0-------to Anode of LED in series with 470 OHM resistor and GND \*/**

//----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ -------

#include<SmartKit.h>

SmartKit KP;//creates a variable of type SmartKit

int LED=0;

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//Baud rate of SmartKit v0.01

void setup(){

KP.SetPort(rx,tx,rst,baud);//Defines the port for communication with SmartKit

KP.PinMode(LED, OUTPUT);

}

void loop(){

KP.DigitalWrite(LED, HIGH);// turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

KP.DigitalWrite(LED, LOW);//turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}//End of loop

//-------------------------------------------------------------- (End) -----------------------------------------------------------------

1. **DigitalRead()**

**Description:** Reads the value from digital pin of SmartKIt either HIGH or a LOW .

**Syntax:** DigitalRead( Pin ) ;

Pin: Digital pin to read.

**Example:** DigitalRead( 4 );

**Returns:** HIGH or LOW**.**

**/\* Ex:2**

**This example demonstrates the use of pinMode(INPUT\_PULLUP). It reads a digital input on pin 4 and prints the results to the serial monitor.**

**The circuit:**

**\* Momentary switch attached from pin 4 to ground**

**\* LED on pin 2**

**With PinMode(Read\_Pin ,INPUT\_PULLUP), an internal 20K-ohm resistor is pulled to 5V. This configuration causes the input to**

**read HIGH when the switch is open, and LOW when it is closed.**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

**D0-------to Anode of LED in series with 470 OHM resistor and GND**

**D4------to Pushbutton and Ground**

\*/

//----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ -------

#include<SmartKit.h>

SmartKit KP;//creates a variable of type SmartKit

int Read\_Pin=4;

int LED=0;

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//Baud rate of SmartKit v0.01

void setup() { Serial.begin(9600);

KP.SetPort(rx,tx,rst,baud);//Defines the port for communication with SmartKit

KP.PinMode(Read\_Pin, INPUT\_PULLUP);//

KP.PinMode(LED, OUTPUT);

}

void loop() {

int sensorVal = KP.DigitalRead(Read\_Pin );//read the pushbutton value into a

//variable

Serial.println(sensorVal); //print out the value of the pushbutton

if (sensorVal == HIGH) {

KP.DigitalWrite(LED, LOW);

}

else {

KP.DigitalWrite(LED, HIGH);

}

}

-------------------------------------------------------------- (End) -----------------------------------------------------------------

1. **AnalogWrite(Pin,Value)**

**Description:** Writes an analog value (PWM) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to AnalogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to AnalogWrite() (or a call to DigitalRead() or DigitalWrite()) on the same pin. The SmartKit has 0,1 ,4,5,6(D0,D1,D4,D5,D6) PWM pins. The frequency of the PWM signal on most pins is approximately 500 Hz.

**Syntax:**AnalogWrite(Pin,Value);

Pin: PWM pin to write to.

Value:the duty cycle between 0 and 255 .

**Example**:AnalogWrite(0,100);

**Returns:** Nothing

**Example Code:**

/\*Ex3

This example shows how to fade an LED using the AnalogWrite() function.

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

**D0-------to Anode of LED and GND (**LED attached from digital pin D0 to ground.)

\*/

//----------------------------------- -(Start )-----------------------------------------------

#include<SmartKit.h>

String name;

SmartKit KP;//creates a variable of type SmartKit

int LED=0;

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//Baud rate of SmartKit v0.01

void setup() {

KP.SetPort(rx,tx,rst,baud);//Defines the port for communication with SmartKit

}

void loop() {

// fade in from min to max in increments of 5 points:

for (int brightness = 0 ; brightness <= 255; brightness += 5) {

KP.AnalogWrite(LED, brightness);

delay(30);

}

// fade out from max to min in increments of 5 points:

for (int brightness = 255 ; brightness >= 0; brightness -= 5)

{

KP.AnalogWrite(LED, brightness);

delay(50);

}

//-------------------------------- (End) ---------------------------------------------------

**Functions for display:**

1. **SetCursor()**

**Description:** Sets the cursor of SmartKit display( 16X2) at column ,row position .

**Syntax:** SetCursor(col,row) where col is column value between 1 and 16 , row is 0 is for first row and 1 is for second row .

**Example:**SetCursor(5,1);

**Returns:**Nothing

1. **Blink():**

**Description:** Displays Blinking cursor .

**Syntax**:Blink()

**Example:** Blink();

**Returns:**Nothing

1. **NoBlink()**

**Description:** opposite to Blink()

**Syntax:** NoBlink()

**Example** NoBlink();

**Returns:** Nothing

1. **Cursor()**

**Description:** Displays cursor an underscore at the position where next character will be written .

**Syntax:** Cursor()

**Example:** Cursor();

**Returns: :** Nothing

1. **NoCursor()**

**Description:** Hides the cursor.

**Syntax:**NoCursor()

**Example**:NoCursor();

**Returns:** Nothing

1. **LcdPrint()**

**Description:** Prints text to display.

**Syntax:**LcdPrint(Data)

Data: the data to print . (int,char,Float or String type )

**Examples**:LcdPrint(01234);

LcdPrint(‘a’);

LcdPrint(“Cavy Agrotronics”);

LcdPrint(123.123456 );

LcdPrint(123.123456 , 2);//Second argument in this function allowed for float value to

//display digits after decimal places ,here it is 2 and will

//display 123.123456 as 123.12

**Returns:** Nothing

**Functions for getting values from alphanumeric keypad :**

1. **GetString()**

**Description:** The cursor starts blinking at the start position (0 , 0) and the Text Mode (**123, abc , ABC, or #$%&** ) is displayed at the Right bottom corner position . The string of max length 80 typed using alphanumeric keypad and send by Enter key is returned to function GetString(). While typing if the length of string exceeds the 15 ,string of 15 characters from end is displayed and if no key pressed for 5 sec scrolling will start to display typed full string until next key press .After key press newly typed character is added at the end of string.

**Syntax:**  GetString( Prompt ) ;//*Prompt is optional*

**Prompt:** Prompt display message on the **Response Screen** of display. Response screen is of length 10 characters from left on the second row of SmartKit display .if the length of prompt is greater than 10 prompt will start scrolling to display whole prompt. String, integer, float type of data is permitted for prompt.

**Example:** GetString( );

**Returns:** String

**Example Code:**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ -------

#include<SmartKit.h>

String name;

SmartKit KP;

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//Baud rate of SmartKit v0.01

void setup(){

KP.SetPort(rx,tx,rst,baud);//

}

void loop(){

name=KP.GetString("Enter your name please");

KP.LcdClear();

KP.LcdPrint("Your name is " + name );

KP.WaitForKeyNum();//Pause until key is pressed

}

-------------------------------------------------------------- (End) -----------------------------------------------------------------

1. **GetIntValue()**

**Description:** Returns the integer value of string entered .

**Syntax:**  GetIntValue( Prompt ) ;//*Prompt is optional*

**Prompt:** Prompt .

**Example:** GetIntValue( );

**Returns:** int

**Example Code:**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ ------------

include<SmartKit.h>

int number;

int sum=15;

SmartKit KP;

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//Baud rate of SmartKit v0.01

void setup(){

KP.SetPort(rx,tx,rst,baud);//

}

void loop(){

KP.LcdClear();

KP.LcdPrint("sum = " + String(sum) );

Delay(2000);

number=KP.GetIntValue("Enter Integer Value ");

sum = sum + number;

KP.LcdClear();

KP.LcdPrint("now sum=" + String(sum) );

Delay(1000);

}

-------------------------------------------------------------- (End) -----------------------------------------------------------------

1. **GetFloatValue()**

**Description:** Returns the Float value of string entered .

**Syntax:**  GetIntValue( Prompt ) ;//Prompt is optional

**Prompt:** Prompt .

**Example:** GetIntValue( );

**Returns:** Float

**Example Code:**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ -------------

#include<SmartKit.h>

SmartKit KP;

int rx=10;

int tx=11;

int rst=13;

long int baud=57600;//

float radius;

void setup() {

KP.SetPort(rx,tx,rst,baud);//

KP.LcdClear();

}

void loop() {

radius=KP.GetFloatValue("Enter radius");

KP.LcdPrint("Area =" + String(radius\*radius\*3.14) );

KP.WaitForKeyNum();//Pause until key is pressed

}

-------------------------------------------------- (End) -----------------------------------------------------------------

1. **WaitForKeyNum()**

**Description:** Waits for key and return the key number of key pressed (1 to 16) .

**Syntax:**  WaitForKeyNum( )

**Example:** WaitForKeyNum( );

**Returns:**int

**Example Code:**

**Connections....**

**Arduino SmartKit**

**pin 13 rst**

**pin 10 tx**

**pin 11 rx**

**+5 v vcc**

**GND GND**

----------------------------------------------------------- -(Start )------------------------------------------------------------------------------ -------------

#include<SmartKit.h>

int num;

String message;

SmartKit KP;

int rx=10;

int tx=11;

int rst=13;

int smartkit\_led;

long int baud=57600;//This is previously defined baud rate of SmartKit v0.01

int val;

void setup() {

KP.SetPort(rx,tx,rst,baud);//

}

void loop() {

num=KP.WaitForKeyNum();

KP.LcdClear();

message=" Key "+String(num)+" pressed";

KP.LcdPrint(message);

}

-------------------------------------------------- (End) -----------------------------------------------------------------