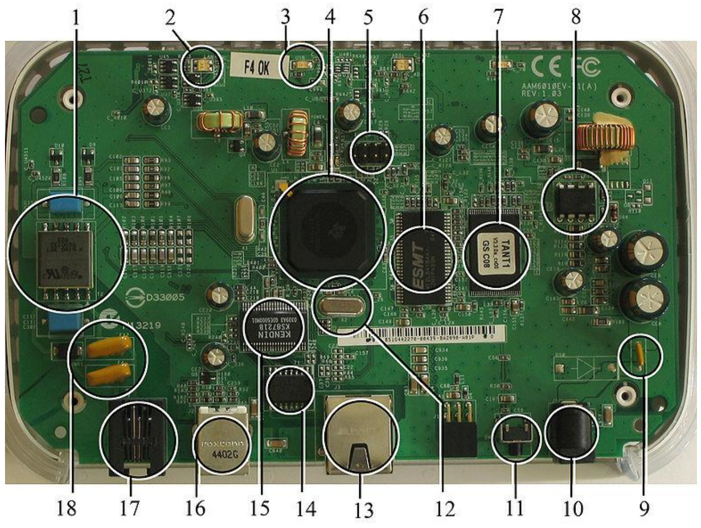
INTRODUCTION

1.1 ABOUT PROGRAMMER'S KEYBOARD:-

Designing and building of a keyboard with those feature which helps in speeding up the programmer or say to coder to do coding on the respective IDE editor window. The client software is standalone, a desktop application.

1.2 Embedded Application



An embedded system is a computer system designed for specific control functions within a larger system, often with real-time computing constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today. Embedded systems contain processing cores that are either microcontrollers or digital signal processors (DSP). A processor is an important unit in the embedded system hardware. It is the heart of the embedded system. The key characteristic, however, is being dedicated to handle a particular task. Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale

1.2.1 History

One of the first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961. It was built from transistor logic and had a hard disk for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad nand gate ICs from $1000/each to $3/each, permitting their use in commercial products. Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. The first microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required many external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

1.2.2 Characteristics

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs. Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.

1.2.3 Processors in Embedded System

Embedded processors can be broken into two broad categories. Ordinary microprocessors (μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have many more peripherals on chip, reducing power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit.

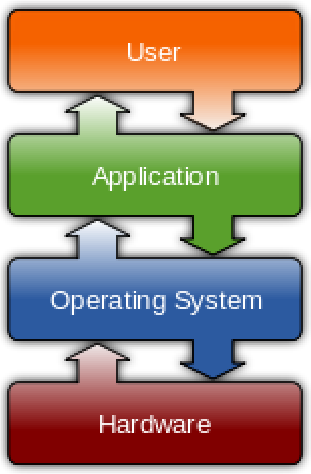
1.3 Firmware

In electronic systems and computing, firmware is the combination of persistent memory and program code and data stored in it. Typical examples of devices containing firmware are embedded systems (such as traffic lights, consumer appliances, and digital watches), computers, computer peripherals, mobile phones, and digital cameras. The firmware contained in these devices provides the control program for the device. Firmware is held in non-volatile memory devices such as ROM, EPROM, or flash memory. Changing the firmware of a device may rarely or never be done during its economic lifetime; some firmware memory devices are permanently installed and cannot be changed after manufacture. Common reasons for updating firmware include fixing bugs or adding features to the device. This may require physically changing ROM integrated circuits, or reprogramming flash memory with a special procedure. Firmware such as the ROM BIOS of a personal computer may contain only elementary basic functions of a device and may only provide services to higher-level software. Firmware such as the program of an embedded system may be the only program that will run on the system and provide all of its functions. Before integrated circuits, other firmware devices included a discrete semiconductor diode matrix. The Apollo guidance computer had firmware consisting of a specially manufactured core memory plane, called "core rope memory", where data was stored by physically threading wires through (1) or around (0) the core storing each data bit

Examples of firmware include:

- Consumer products:
  - Timing and control systems for washing machines
  - Controlling sound and video attributes, as well as the channel list, in modern TVs
  - EPROM chips used in the Eventide H-3000 series of digital music processors
  - In computers
  - The BIOS found in IBM-compatible personal computers.
  - The platform code found on Itanium systems, Intel-based Mac OS X machines, and many Intel desktop boards is EFI-compliant firmware.
  - Open Firmware, used in computers from Sun Microsystems, Apple, and Genesi.
  - ARCS (computing), used in computers from Silicon Graphics.
  - Kickstart, used in the Amiga line of computers (POST, hardware in it + Plug and Play auto-configuration of peripherals, kernel, etc.).
  - RTAS (Run-Time Abstraction Services), used in computers from IBM.

1.4 Device Drive Software



In computing, a device driver is a computer program that operates or controls a particular type of device that is attached to a computer. A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects.When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware-dependent and operating-system-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface.

1.4.1 Purpose of device drive software

A device driver simplifies programming by acting as translator between a hardware device and the applications or operating systems that use it.[1] Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using. Physical layers communicate with specific device instances. For example, a serial port needs to handle standard communication protocols such as XON/XOFF that are common for all serial port hardware. This would be managed by a serial port logical layer. However, the physical layer needs to communicate with a particular serial port chip. 16550 UART hardware differs from PL-011. The physical layer addresses these chip-specific variations. Conventionally, OS requests go to the logical layer first. In turn, the logical layer calls upon the physical layer to implement OS requests in terms understandable by the hardware. Conversely, when a hardware device needs to respond to the OS, it uses the physical layer to speak to the logical layer. In Linux environments, programmers can build device drivers either as parts of the kernel or separately as loadable modules. Make dev includes a list of the devices in Linux: ttyS (terminal), lp (parallel port), hd (disk), loop (loopback disk device), sound (these include mixer, sequencer, dsp, and audio).The Microsoft Windows .sys files and Linux .ko modules contain loadable device drivers. The advantage of loadable device drivers is that they can be loaded only when necessary and then unloaded, thus saving kernel memory.

1.4.2 Development

Writing a device driver requires an in-depth understanding of how the hardware and the software of a given platform function. Drivers operate in a highly privileged environment and can cause disaster if they get things wrong. In contrast, most user-level software on modern operating systems can be stopped without greatly affecting the rest of the system. Even drivers executing in user mode can crash a system if the device is erroneously programmed. These factors make it more difficult and dangerous to diagnose problems.

The task of writing drivers thus usually falls to software engineers or computer engineers who work for hardware-development companies. This is because they have better information than most outsiders about the design of their hardware. Moreover, it was traditionally considered in the hardware manufacturer's interest to guarantee that their clients can use their hardware in an optimum way. Typically, the logical device driver (LDD) is written by the operating system vendor, while the physical device driver (PDD) is implemented by the device vendor. But in recent years non-vendors have written numerous device drivers, mainly for use with free and open source operating systems. In such cases, it is important that the hardware manufacturer provides information on how the device communicates. Although this information can instead be learned by reverse engineering, this is much more difficult with hardware than it is with software.

Microsoft has attempted to reduce system instability due to poorly written device drivers by creating a new framework for driver development, called Windows Driver Foundation (WDF). This includes User-Mode Driver Framework (UMDF) that encourages development of certain types of drivers—primarily those that implement a message-based protocol for communicating with their devices—as user-mode drivers. If such drivers malfunction, they do not cause system instability. The Kernel-Mode Driver Framework (KMDF) model continues to allow development of kernel-mode device drivers, but attempts to provide standard implementations of functions that are known to cause problems, including cancellation of I/O operations, power management, and plug and play device support. Apple has an open-source framework for developing drivers on Mac OS X called the I/O Kit.

1.4.3 Kernel mode vs. User mode

Device drivers, particularly on modern Microsoft Windows platforms, can run in kernel-mode (Ring 0 on x86 CPUs) or in user-mode (Ring 3 on x86 CPUs). The primary benefit of running a driver in user mode is improved stability, since a poorly written user mode device driver cannot crash the system by overwriting kernel memory. On the other hand, user/kernel-mode transitions usually impose a considerable performance overhead, thereby prohibiting user-mode drivers for low latency and high throughput requirements.

Kernel space can be accessed by user module only through the use of system calls. End user programs like the UNIX shell or other GUI-based applications are part of the user space. These applications interact with hardware through kernel supported functions.

1.4.4 Application

Because of the diversity of modern hardware and operating systems, drivers operate in many different environments.[7] Drivers may interface with:

- printers
- video adapters
- Network cards
- Sound cards
- Local buses of various sorts—in particular, for bus mastering on modern systems
- Low-bandwidth I/O buses of various sorts (for pointing devices such as mice, keyboards, USB, etc.)
- Computer storage devices such as hard disk, CD-ROM, and floppy disk buses (ATA, SATA, SCSI)
- Implementing support for different file systems
- Image scanners
- Digital cameras

Common levels of abstraction for device drivers include:

- For hardware:
    - Interfacing directly
    - Writing to or reading from a device control register
    - Using some higher-level interface (e.g. Video BIOS)
    - Using another lower-level device driver (e.g. file system drivers using disk drivers)
    - Simulating work with hardware, while doing something entirely different.
- For software:
    - Allowing the operating system direct access to hardware resources
    - Implementing only primitives
    - Implementing an interface for non-driver software (e.g., TWAIN)
    - Implementing a language, sometimes quite high-level (e.g., PostScript)

Choosing and installing the correct device drivers for given hardware is often a key component of computer system configuration.

1.4.5 Identifiers

A device on the PCI bus or USB is identified by two IDs which consist of 4 hexadecimal numbers each. The vendor ID identifies the vendor of the device. The device ID identifies a specific device from that manufacturer/vendor.

A PCI device has often an ID pair for the main chip of the device, and also a subsystem ID pair which identifies the vendor, which may be different from the chip manufacturer.

1.5 Purpose of the Project

This project is partly based on embedded systems and rest of the part is software based. Special key sets are provided which are dedicated to do specific task. This project is been developed keeping the programmers as our audience. For now, this will be developed for windows platform. This project will not be developed as per IEEE, FCC (Federal Communication Commission) Electronics specification. This keyboard will have a USB interface to connect with your system. Software will be developed further for rendering this keyboard from your system. That software will reside on computer system.

1.6 Benefits of the Project

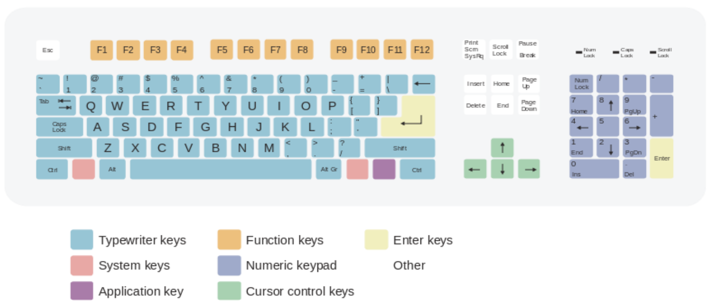
- This project has been specifically designed for the programmers to speed up their coding process.
- This project is useful for the programmer which works on the different platforms.

Literature Review

2.1 COMPUTER KEYBOARD

In computing, a keyboard is a typewriter-style device, which uses an arrangement of buttons or keys, to act as mechanical levers or electronic switches. Following the decline of punch cards and paper tape, interaction via teleprinter-style keyboards became the main input device for computers.

A keyboard typically has characters engraved or printed on the keys and each press of a key typically corresponds to a single written symbol. However, to produce some symbols requires pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce letters, numbers or signs (characters), other keys or simultaneous key presses can produce actions or computer commands.



Despite the development of alternative input devices, such as the mouse, touchscreen, pen devices, character recognition and voice recognition, the keyboard remains the most commonly used and most versatile device used for direct (human) input into computers.

In normal usage, the keyboard is used to type text and numbers into a word processor, text editor or other programs. In a modern computer, the interpretation of key presses is generally left to the software. A computer keyboard distinguishes each physical key from every other and reports all key presses to the controlling software. Keyboards are also used for computer gaming, either with regular keyboards or by using keyboards with special gaming features, which can expedite frequently used keystroke combinations. A keyboard is also used to give commands to the operating system of a computer, such as Windows' Control-Alt-Delete combination, which brings up a task window or shuts down the machine. Keyboards are the only way to enter commands on a command-line interface.

2.1.1 Computer Keyboard History

While typewriters are the definitive ancestor of all key-based text entry devices, the computer keyboard as a device for electromechanical data entry and communication derives largely from the utility of two devices: teleprinter (or teletypes) and keypunches. It was through such devices that modern computer keyboards inherited their layouts.

As early as the 1870s, teleprinter-like devices were used to simultaneously type and transmit stock market text data from the keyboard across telegraph lines to stock ticker machines to be immediately copied and displayed onto ticker tape. The teleprinter, in its more contemporary form, was developed from 1903–1910 by American mechanical engineer Charles Krum and his son Howard, with early contributions by electrical engineer Frank Pearne. Earlier models were developed separately by individuals such as Royal Earl House and Frederick G. Creed.

Earlier, Herman Hollerith developed the first keypunch devices, which soon evolved to include keys for text and number entry akin to normal typewriters by the 1930s.

The keyboard on the teleprinter played a strong role in point-to-point and point-to-multipoint communication for most of the 20th century, while the keyboard on the keypunch device played a strong role in data entry and storage for just as long. The development of the earliest computers incorporated electric typewriter keyboards: the development of the ENIAC computer incorporated a keypunch device as both the input and paper-based output device, while the BINAC computer also made use of an electromechanically controlled typewriter for both data entry onto magnetic tape (instead of paper) and data output.

2.1.2 Keyboard Types

One factor determining the size of a keyboard is the presence of duplicate keys, such as a separate numeric keyboard, for convenience.

Further the keyboard size depends on the extent to which a system is used where a single action is produced by a combination of subsequent or simultaneous keystrokes (with modifier keys, see below), or multiple pressing of a single key. A keyboard with few keys is called a keypad. See also text entry interface.

Another factor determining the size of a keyboard is the size and spacing of the keys. Reduction is limited by the practical consideration that the keys must be large enough to be easily pressed by fingers. Alternatively a tool is used for pressing small keys.

2.1.2.1 Standard

Standard "full-travel" alphanumeric keyboards have keys that are on three-quarter inch centres (0.750 inches, 19.05 mm), and have a key travel of at least 0.150 inches (3.81 mm). Desktop computer keyboards, such as the 101-key US traditional keyboards or the 104-key Windows keyboards, include alphabetic characters, punctuation symbols, numbers and a variety of function keys. The internationally common 102/105 key keyboards have a smaller left shift key and an additional key with some more symbols between that and the letter to its right (usually Z or Y). Also the enter key is usually shaped differently. Computer keyboards are similar to electric-typewriter keyboards but contain additional keys. Standard USB keyboards can also be connected to some non-desktop devices.

2.1.2.2 Laptop Size

Keyboards on laptops and notebook computers usually have a shorter travel distance for the keystroke and a reduced set of keys. They may not have a numerical keypad, and the function keys may be placed in locations that differ from their placement on a standard, full-sized keyboard.

2.1.2.3 Thumb-sized

Smaller external keyboards have been introduced for devices without a built-in keyboard, such as PDAs, and smartphones. Small keyboards are also useful where there is a limited workspace.

A chorded keyboard allows pressing several keys simultaneously. For example, the GKOS keyboard has been designed for small wireless devices. Other two-handed alternatives more akin to a game controller, such as the Alpha Grip, are also used as a way to input data and text.

A thumb keyboard (thumb board) is used in some personal digital assistants such as the Palm Treo and BlackBerry and some Ultra-Mobile PCs such as the OQO.

Numeric keyboards contain only numbers, mathematical symbols for addition, subtraction, multiplication, and division, a decimal point, and several function keys. They are often used to facilitate data entry with smaller keyboards that do not have a numeric keypad, commonly those of laptop computers. These keys are collectively known as a numeric pad, numeric keys, or a numeric keypad, and it can consist of the following types of keys:-

- Arithmetic operators such as +, -, *, /
- Numeric digits 0–9
- Cursor arrow keys
- Navigation keys such as Home, End, PgUp, PgDown, etc.
- Num Lock button, used to enable or disable the numeric pad
- Enter key

2.1.3 Non-standard layout and special-use types

2.1.3.1 Chorded

While other keyboards generally associate one action with each key, chorded keyboards associate actions with combinations of key presses. Since there are many combinations available, chorded keyboards can effectively produce more actions on a board with fewer keys. Court reporters' stenotype machines use chorded keyboards to enable them to enter text much faster by typing a syllable with each stroke instead of one letter at a time. The fastest typists (as of 2007) use a stenograph, a kind of chorded keyboard used by most court reporters and closed-caption reporters. Some chorded keyboards are also made for use in situations where fewer keys are preferable, such as on devices that can be used with only one hand, and on small mobile devices that don't have room for larger keyboards. Chorded keyboards are less desirable in many cases because it usually takes practice and memorization of the combinations to become proficient.

2.1.3.2 Software

Software keyboards or on-screen keyboards often take the form of computer programs that display an image of a keyboard on the screen. Another input device such as a mouse or a touchscreen can be used to operate each virtual key to enter text. Software keyboards have become very popular in touchscreen enabled cell phones, due to the additional cost and space requirements of other types of hardware keyboards. Microsoft Windows, Mac OS X, and some varieties of Linux include on-screen keyboards that can be controlled with the mouse.

2.1.3.3 Foldable

Further information: Flexible electronics.

Foldable (also called flexible) keyboards are made of soft plastic or silicone which can be rolled or folded on itself for travel. [3] When in use, these keyboards can conform to uneven surfaces, and are more resistant to liquids than standard keyboards. These can also be connected to portable devices and smartphones. Some models can be fully immersed in water, making them popular in hospitals and laboratories, as they can be disinfected.

2.1.3.4 Projection (as by laser)

Projection keyboards project an image of keys, usually with a laser, onto a flat surface. The device then uses a camera or infrared sensor to "watch" where the user's fingers move, and will count a key as being pressed when it "sees" the user's finger touch the projected image. Projection keyboards can simulate a full size keyboard from a very small projector. Because the "keys" are simply projected images, they cannot be felt when pressed. Users of projected keyboards often experience increased discomfort in their fingertips because of the lack of "give" when typing. A flat, non-reflective surface is also required for the keys to be projected. Most projection keyboards are made for use with PDAs and smartphones due to their small form factor.

2.1.3.5 Optical keyboard technology

Also known as photo-optical keyboard, light responsive keyboard, photo-electric keyboard and optical key actuation detection technology.

An optical keyboard technology utilizes light emitting devices and photo sensors to optically detect actuated keys. Most commonly the emitters and sensors are located in the perimeter, mounted on a small PCB. The light is directed from side to side of the keyboard interior and it can only be blocked by the actuated keys. Most optical keyboards require at least 2 beams (most commonly vertical beam and horizontal beam) to determine the actuated key. Some optical keyboards use a special key structure that blocks the light in a certain pattern, allowing only one beam per row of keys (most commonly horizontal beam).

2.1.4 Layout

2.1.4.1 Alphabetic

There are a number of different arrangements of alphabetic, numeric, and punctuation symbols on keys. These different keyboard layouts arise mainly because different people need easy access to different symbols, either because they are inputting text in different languages, or because they need a specialized layout for mathematics, accounting, computer programming, or other purposes. The United States keyboard layout is used as default in the currently most popular operating systems: Windows, [4] Mac OS X [5] and Linux. [6][7] The common QWERTY-based layout was designed early in the era of mechanical typewriters, so its ergonomics were compromised to allow for the mechanical limitations of the typewriter.

As the letter-keys were attached to levers that needed to move freely, inventor Christopher Sholes developed the QWERTY layout to reduce the likelihood of jamming. With the advent of computers, lever jams are no longer an issue, but nevertheless, QWERTY layouts were adopted for electronic keyboards because they were widely used. Alternative layouts such as the Dvorak Simplified Keyboard are not in widespread use.

The QWERTZ layout is widely used in Germany and much of Central Europe. The main difference between it and QWERTY is that Y and Z are swapped, and most special characters such as brackets are replaced by diacritical characters.

Another situation takes place with "national" layouts. Keyboards designed for typing in Spanish have some characters shifted, to release the space for Ñ; similarly, those for French and other European languages may have a special key for the character Ç. The AZERTY layout is used in France, Belgium and some neighbouring countries. It differs from the QWERTY layout in that the A and Q are swapped, the Z and Ware swapped, and the M is moved from the right of N to the right of L (where colon/semicolon is on a US keyboard). The digits 0 to 9 are on the same keys, but to be typed the shift key must be pressed. The upshifted positions are used for accented characters. Keyboards in many parts of Asia may have special keys to switch between the Latin character set and a completely different typing system. Japanese layout keyboards can be switched between various Japanese input methods and the Latin alphabet by signalling the operating system's input interpreter of the change, and some operating systems (namely the Windows family) interpret the character "\" as "¥" for display purposes without changing the byte code which has led some keyboard makers to mark "\" as "¥" or both. In the Arab world, keyboards can often be switched between Arabic and Latin characters.

In bilingual regions of Canada and in the French-speaking province of Québec, keyboards can often be switched between English and a French-language keyboard; while both keyboards share the same QWERTY alphabetic layout, the French-language keyboard enables the user to type accented vowels such as "é" or "à" with a single keystroke. Using keyboards for other languages leads to a conflict: the image on the key does not correspond to the character. In such cases, each new language may require an additional label on the keys, because the standard keyboard layouts do not share even similar characters of different languages (see the example in the figure above).

2.1.5 Key types

2.1.5.1 Alphanumeric

Alphabetical, numeric, and punctuation keys are used in the same fashion as a typewriter keyboard to enter their respective symbol into a word processing program, text editor, data spread sheet, or other program. Many of these keys will produce different symbols when modifier keys or shift keys are pressed. The alphabetic characters become uppercase when the shift key or Caps Lock key is depressed. The numeric characters become symbols or punctuation marks when the shift key is depressed. The alphabetical, numeric, and punctuation keys can also have other functions when they are pressed at the same time as some modifier keys.

The Space bar is a horizontal bar in the lowermost row, which is significantly wider than other keys. Like the alphanumeric characters, it is also descended from the mechanical typewriter. Its main purpose is to enter the space between words during typing. It is large enough so that a thumb from either hand can use it easily. Depending on the operating system, when the space bar is used with a modifier key such as the control key, it may have functions such as resizing or closing the current window, half-spacing, or backspacing. In computer games and other applications the key has myriad uses in addition to its normal purpose in typing, such as jumping and adding marks to check boxes. In certain programs for playback of digital video, the space bar is used for pausing and resuming the playback.

2.1.5.2 Modifiers

Modifier keys are special keys that modify the normal action of another key, when the two are pressed in combination. For example, <Alt> + <F4> in Microsoft Windows will close the program in an active window. In contrast, pressing just <F4> will probably do nothing, unless assigned a specific function in a particular program. By themselves, modifier keys usually do nothing.

The most widely used modifier keys include the Control key, Shift key and the Alt key. The AltGr key is used to access additional symbols for keys that have three symbols printed on them. On the Macintosh and Apple keyboards, the modifier keys are the Option key and Command key, respectively. On MIT computer keyboards, the Meta key is used as a modifier and for Windows keyboards, there is a Windows key. Compact keyboard layouts often use a Fn key. "Dead keys" allow placement of a diacritic mark, such as an accent, on the following letter (e.g., the Compose key).

The Enter/Return key typically causes a command line, window form or dialog box to operate its default function, which is typically to finish an "entry" and begin the desired process. In word processing applications, pressing the enter key ends a paragraph and starts a new one.

2.1.5.3 Navigation and typing modes

Navigation keys include a variety of keys which move the cursor to different positions on the screen. Arrow keys are programmed to move the cursor in a specified direction; page scroll keys, such as the Page Up and Page Down keys, scroll the page up and down. The Home key is used to return the cursor to the beginning of the line where the cursor is located; the End key puts the cursor at the end of the line. The Tab key advances the cursor to the next tab stop.

The Insert key is mainly used to switch between overtype mode, in which the cursor overwrites any text that is present on and after its current location, and insert mode, where the cursor inserts a character at its current position, forcing all characters past it one position further. The Delete key discards the character ahead of the cursor's position, moving all following characters one position "back" towards the freed place. On many notebook computer keyboards the key labelled Delete (sometimes Delete and Backspace are printed on the same key) serves the same purpose as a Backspace key. The Backspace key deletes the preceding character.

Lock keys lock part of a keyboard, depending on the settings selected. The lock keys are scattered around the keyboard. Most styles of keyboards have three LEDs indicating which locks are enabled, in the upper right corner above the numeric pad. The lock keys include Scroll lock, Num lock (which allows the use of the numeric keypad), and Caps lock.

2.1.5.4 System commands

The SysRq and Print screen commands often share the same key. SysRq was used in earlier computers as a "panic" button to recover from crashes. The Print screen command used to capture the entire screen and send it to the printer, but in the present it usually puts a screenshot in the clipboard. The Break key/Pause key no longer has a well-defined purpose. Its origins go back to teleprinter users, who wanted a key that would temporarily interrupt the communications line. The Break key can be used by software in several different ways, such as to switch between multiple login sessions, to terminate a program, or to interrupt a modem connection.

In programming, especially old DOS-style BASIC, Pascal and C, Break is used (in conjunction with Ctrl) to stop program execution. In addition to this, Linux and variants, as well as many DOS programs, treat this combination the same as Ctrl+C. On modern keyboards, the break key is usually labelled Pause/Break. In most Windows environments, the key combination Windows key Pause brings up the system properties.

The Escape key (often abbreviated Esc) is used to initiate an escape sequence. As most computer users no longer are concerned with the details of controlling their computer's peripherals, the task for which the escape sequences were originally designed, the escape key was appropriated by application programmers, most often to "escape" or back out of a mistaken command. This use continues today in Microsoft Windows's use of escape as a shortcut in dialog boxes for No, Quit, Exit, Cancel, or Abort.

2.1.5.5 Miscellaneous

Many, but not all, computer keyboards have a numeric keypad to the right of the alphabetic keyboard which contains numbers, basic mathematical symbols (e.g., addition, subtraction, etc.), and a few function keys. On Japanese/Korean keyboards, there may be Language input keys. Some keyboards have power management keys (e.g., power key, sleep key and wake key); Internet keys to access a web browser or E-mail; and/or multimedia keys, such as volume controls or keys that can be programmed by the user to launch a specified software or command like launching a game or minimize all windows.

2.1.5.6 Multiple layouts

It is possible to install multiple keyboard layouts within an operating system and switch between them, either through features implemented within the OS, or through an external application. Microsoft Windows, [8] Linux, [9] and Mac [10] provide support to add keyboard layouts and choose from them.

2.1.5.7 Layout changing software

The character code produced by any key press is determined by the keyboard driver software. A key press generates a scan code which is interpreted as an alphanumeric character or control function. Depending on operating systems, various application programs are available to create, add and switch among keyboard layouts. Many programs are available, some of which are language specific.

The arrangement of symbols of specific language can be customized. An existing keyboard layout can be edited, and a new layout can be created using this type of software.

A common application today of the Esc key is as a shortcut key for the Stop button in many web browsers. On machines running Microsoft Windows, prior to the implementation of the Windows key on keyboards, the typical practice for invoking the "start" button was to hold down the control key and press escape. This process still works in Windows 2000, XP, Windows Vista and Windows 7.

The Menu key or Application key is a key found on Windows-oriented computer keyboards. It is used to launch a context menu with the keyboard rather than with the usual right mouse button. The key's symbol is a small icon depicting a cursor hovering above a menu. This key was created at the same time as the Windows key. This key is normally used when the right mouse button is not present on the mouse. Some Windows public terminals do not have a Menu key on their keyboard to prevent users from right-clicking (however, in many windows applications, a similar functionality can be invoked with the Shift+F10 keyboard shortcut).

For example, Ukelele for Mac, The Microsoft Keyboard Layout Creator and open-source Avro Keyboard for Windows provide the ability to customize the keyboard layout as desired. Other programs with similar functions include Avro Keyboard, Tavultesoft Keyman Developer, The Microsoft Keyboard Layout Creator, Mount Focus Keyboard Designer, Map Keyboard, KbdEdit, Key Customizer, Keyboard Remapper, Infine Keyboard Commander for Windows; and X Neural Switcher, Keyboard Layout Editor and Keyboard Layout Creator for Linux.

2.1.6 Technology

2.1.6.1 Key switches

In the first electronic keyboards in the early 1970s, the key switches were individual switches inserted into holes in metal frames. These keyboards cost from 80–120 US dollars and were used in mainframe data terminals. The most popular switch types were reed switches (contacts enclosed in a vacuum in a glass capsule, affected by a magnet mounted on the switch plunger – from Clare-Pendar in Post Falls Idaho, which became part of General Instrument, which used reed switch capsules made by C.P. Clare Co. in Illinois; and Key Tronic Corporation of Spokane, Washington), Hall-effect switches (using a Hall-effect semiconductor where a current is generated by a passing magnet – from Micro switch [19] in Illinois, which became part of Honeywell), and inductive core switches (again, activated by a magnet – from Cortron,which was part of ITW/Illinois Tool Works). These switches were rated to last for 100 million cycles and had 0.187-inch (4.75 mm) key travel, compared to 0.110 inch (2.79 mm) today.

In the mid-1970s, lower-cost direct-contact key switches were introduced, but their life in switch cycles was much shorter (rated ten million cycles) because they were open to the environment. This became more acceptable, however, for use in computer terminals at the time, which began to see increasingly shorter model lifespans as they advanced.

In 1978, Key Tronic Corporation introduced keyboards with capacitive-based switches, one of the first keyboard technologies to not use self-contained switches. There was simply a sponge pad with a conductive-coated Mylar plastic sheet on the switch plunger, and two half-moon trace patterns on the printed circuit board below. As the key was depressed, the capacitance between the plunger pad and the patterns on the PCB below changed, which was detected by integrated circuits (IC). These keyboards were claimed to have the same reliability as the other "solid-state switch" keyboards such as inductive and Hall-Effect, but competitive with direct-contact keyboards. Prices of $60 for keyboards were achieved and Key Tronic rapidly became the largest independent keyboard manufacturer.

Meanwhile, IBM made their own keyboards, using their own patented technology: Keys on older IBM keyboards were made with a "buckling spring" mechanism, in which a coil spring under the key buckles under pressure from the user's finger, triggering a hammer that presses two plastic sheets (membranes) with conductive traces together, completing a circuit. This produces a clicking sound, and gives physical feedback for the typist indicating that the key has been depressed.

The first electronic keyboards had a typewriter key travel distance of 0.187 inches (4.75 mm), key tops were a half-inch (12.7 mm) high, and keyboards were about two inches (5 cm) thick. Over time, less key travel was accepted in the market, finally landing on 0.110 inches (2.79 mm). Coincident with this, Key Tronic was the first company to introduce a keyboard which was only about one inch thick. And now keyboards measure only about a half-inch thick.

Three final mechanical technologies brought keyboards to where they are today, driving the cost well under $10:

- "Monoblock" keyboard designs were developed where individual switch housings were eliminated and a one-piece "monoblock" housing used instead. This was possible because of molding techniques that could provide very tight tolerances for the switch-plunger holes and guides across the width of the keyboard so that the key plunger-to-housing clearances were not too tight or too loose, either of which could cause the keys to bind.
- The use of contact-switch membrane sheets under the monoblock. This technology came from flat-panel switch membranes, where the switch contacts are printed inside of a top and bottom layer, with a spacer layer in between, so that when pressure is applied to the area above, a direct electrical contact is made. The membrane layers can be printed by very-high volume, low-cost "reel-to-reel" printing machines, with each keyboard membrane cut and punched out afterwards.
- The use of pad-printed keytops (called "Tampo printed" at the time because Tampo [23] was the most popular equipment manufacturer). Initially sublimation ink was used (see above), but very durable clear-coats are now printed over the key legends to protect them. These coatings are also used to reduce glare, and in many cases have an anti-microbial content added for user protection.

Plastic materials played a very important part in the development and progress of electronic keyboards. Until "monoblocks" came along, GE's "self-lubricating" Delrin was the only plastic material for keyboard switch plungers that could withstand the beating over tens of millions of cycles of lifetime use. Greasing or oiling switch plungers was undesirable because it would attract dirt over time which would eventually affect the feel and even bind the key switches (although keyboard manufacturers would sometimes sneak this into their keyboards, especially if they could not control the tolerances of the key plungers and housings well enough to have a smooth key depression feel or prevent binding). But Delrin was only available in black and white, and was not suitable for keytops (too soft), so keytops use ABS plastic. However, as plastic molding advanced in maintaining tight tolerances, and as key travel length reduced from 0.187-inch to 0.110-inch (4.75 mm to 2.79 mm), single-part keytop/plungers could be made of ABS, with the keyboard monolocks also made of ABS.

2.1.6.2 Control processor

Computer keyboards include control circuitry to convert key presses into key codes (usually scancodes) that the computer's electronics can understand. The key switches are connected via the printed circuit board in an electrical X-Y matrix where a voltage is provided sequentially to the Y lines and, when a key is depressed, detected sequentially by scanning the X lines.

The first computer keyboards were for mainframe computer data terminals and used discrete electronic parts. The first keyboard microprocessor was introduced in 1972 by General Instruments, but keyboards have been using the single-chip 8048 microcontroller variant since it became available in 1978. The keyboard switch matrix is wired to its inputs, it converts the keystrokes to key codes, and, for a detached keyboard, sends the codes down a serial cable (the keyboard cord) to the main processor on the computer motherboard. This serial keyboard cable communication is only bi-directional to the extent that the computer's electronics controls the illumination of the caps lock, num lock and scroll lock lights.

One test for whether the computer has crashed is pressing the caps lock key. The keyboard sends the key code to the keyboard driver running in the main computer; if the main computer is operating, it commands the light to turn on. All the other indicator lights work in a similar way. The keyboard driver also tracks the Shift, alt and control state of the keyboard.

2.1.6.3 Connection types

There are several ways of connecting a keyboard to a system unit (more precisely, to its keyboard controller) using cables, including the standard AT connector commonly found on motherboards, which was eventually replaced by the PS/2 and the USB connection. Prior to the iMac line of systems, Apple used the proprietary Apple Desktop Bus for its keyboard connector.

Wireless keyboards have become popular for their increased user freedom. A wireless keyboard often includes a required combination transmitter and receiver unit that attaches to the computer's keyboard port. The wireless aspect is achieved either by radio frequency (RF) or by infrared (IR) signals sent and received from both the keyboard and the unit attached to the computer. A wireless keyboard may use an industry standard RF, called Bluetooth. With Bluetooth, the transceiver may be built into the computer. However, a wireless keyboard needs batteries to work and may pose a security problem due to the risk of data "eavesdropping" by hackers. Wireless solar keyboards charge their batteries from small solar panels using sunlight or standard artificial lighting. An early example of a consumer wireless keyboard is that of the Olivetti Envision.

2.1.7 Other issues

2.1.7.1 Keystroke logging

Keystroke logging (often called keylogging) is a method of capturing and recording user keystrokes. While it is used legally to measure employee productivity on certain clerical tasks, or by law enforcement agencies to find out about illegal activities, it is also used by hackers for various illegal or malicious acts. Hackers use keyloggers as a means to obtain passwords or encryption keys and thus bypass other security measures.

Keystroke logging can be achieved by both hardware and software means. Hardware key loggers are attached to the keyboard cable or installed inside standard keyboards. Software keyloggers work on the target computer's operating system and gain unauthorized access to the hardware, hook into the keyboard with functions provided by the OS, or use remote access software to transmit recorded data out of the target computer to a remote location. Some hackers also use wireless keylogger sniffers to collect packets of data being transferred from a wireless keyboard and its receiver, and then they crack the encryption key being used to secure wireless communications between the two devices. Anti-spyware applications are able to detect many keyloggers and cleanse them. Responsible vendors of monitoring software support detection by anti-spyware programs, thus preventing abuse of the software. Enabling a firewall does not stop keyloggers per se, but can possibly prevent transmission of the logged material over the net if properly configured. Network monitors (also known as reverse-firewalls) can be used to alert the user whenever an application attempts to make a network connection. This gives the user the chance to prevent the keylogger from "phoning home" with his or her typed information. Automatic form-filling programs can prevent keylogging entirely by not using the keyboard at all. Most keyloggers can be fooled by alternating between typing the login credentials and typing characters somewhere else in the focus window.

2.1.8 Working of Keyboard

2.1.8.1 Functioning of a Computer Keyboard

In general, there are 80-110 keys in a computer keyboard. The keys may vary depending upon the brand and the type of operating system. Nevertheless, the shape, size and spacing of keys are almost same for all keyboards. Also the layout or arrangement of keys that represent letters, signs and symbols is same, which is referred to as QWERTY.

The working of a computer keyboard can be compared to a miniature computer. Inside the keyboard, there are metallic plate, circuit board (key matrix) and processor, which are responsible for transferring information from the keyboard to the computer. Depending upon the working principle, there are two main types of keys, namely, capacitive and hard-contact. Let's discuss in brief about the functioning of capacitive and hard contact key.

2.1.8.2 Capacitive Key

On the underside of a capacitive key, a metal plunger is fixed, which helps in activating the circuit flow. When a capacitive key is pressed, the metal plunger applies a gentle pressure to the circuit board. The pressure is identified by the computer and the circuit flow is initiated, resulting in the transfer of information from the circuit to the currently installed software.

2.1.8.3 Hard Contact Key

A hard contact key is attached with a metallic plate that helps in connecting the circuit board. When the hard contact key is pressed, it pushes a metallic plate, which in turn touches the metallic portion of the circuit plate. This overall process of completing a circuit results in a circuit flow, allowing the transfer of the message to the central processing unit (CPU), which is further transmitted to the software. In both the key types, the circuit signals the processor to read and/or identify the character that has been pressed. For example, in a hard contact key, the processor reads that pressing 'shift' and 'a' keys at the same time corresponds to 'A'. Hence accordingly, the letter, sign or symbol is displayed on the screen. Releasing the pressed key breaks the circuit flow, after which the key retains its original position. The communication between a computer keyboard and main computer is bi-directional, meaning that message or information can be sent within each other.

2.1.8.4 Layout

Computer keyboards are an input device. They put the information a person types into a program on the computer. Most keyboards have 80 to 110 keys. The numbers and letters on the keyboard are displayed keycaps--these are the buttons that are pressed when a person types. The layout of the numbers and letters are the same on every keyboard and they are referred to as the QWERTY.

2.1.8.5 Key matrix

The inside of the keyboard is like a mini-computer and consists of a processor and circuits. These transfer the information to the processor inside of the computer. Inside of the keyboard's processor resides the key matrix. The key matrix is a grid of circuits. These circuits are individually placed under each key. When a key is pushed, it pushes the switch on the circuit board underneath the key causing an electrical current to pass through the circuit and into the processor. When the current passes through, the switch vibrates, signalling the processor to read it.

2.1.8.6 Keymap

The circuit is closed when a key is pressed. The closing of the circuit signals the processor to read the Keymap stored within it. The processor uses the Keymap, sometimes called the character map, to find the key that is closed off on the board. By using the Keymap, the processor in the keyboard can tell which letter is being pushed and if it should be a capital or lowercase letter depending on if the shift key is being pressed.

2.1.8.7 Communication

The keyboard connects to the computer via a five pin male plug or a PS/2 plug. Keyboards and computers work together in a bi-directional format. This means that they can each send information to one another. These bi-directional lines are the clock line coming from the keyboard and the data line coming from the computer. Both lines must be idle, or high in order for the keyboard to send data. The computer will send a signal to the keyboard through the clock line letting it know that the line is clear to send. If the line is not clear, the keyboard will hold the information until the line opens. When the line is low, the keyboard is waiting for a command from the computer. When the computer wants to send information to the keyboard, it brings the data and the clock line low. It does this to ensure that the keyboard does not send it a message at the same time.

2.2 Serial Communication

2.2.1 Introduction

Whoever work with standard RS232 components from a .NET environment, and need real, embedded-like RS232 communication, realize in a short time that the component needs to be wrapped, or at least, additionally supported by custom software. I designed this tutorial because I couldn’t find something similar on the web. The code is an excerpt from a huge GPS tracking application, and it proved to work nicely. Just to mention, that there are other methods for RS232 communication such as using hand-shaking protocols, and hardware/software enabled pin control.

2.2.2 Background

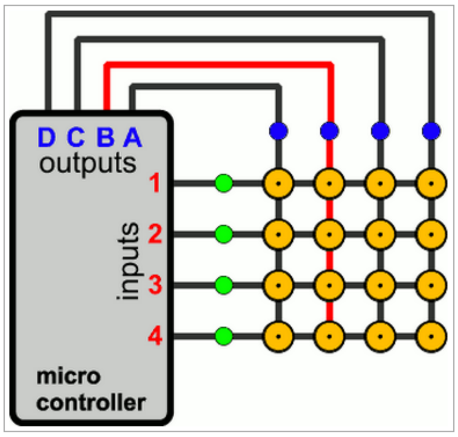
I won’t go in to the details of basic RS232 communication, except that it is a point-to-point communication between two hosts, and they could be computers or embedded devices. My point here will be the software implementation part, using C# in a .NET 2.0 environments to be precise, but the principles are adoptable to any other language and IDE. Also, the type of data I’m going to send/read is string, but with little more effort, the code can be adopted to send/receive any type of serializable data. One more restriction, the demo code is adopted to receive a huge block of data and then process it. Another way is to continually append read data to some FIFO buffer, and do an online parsing of whatever data has arrived at the time (search for delimiters inside the arrived data before the reading ends), for continual receiving of data without any pause between two packets.

3. Problem Identification

3.1 Determining the Problem

Firstly, here we may define statements to define the problems which may be faced or which are standing before us. As we determine and frame the problems, many new topics will be discussed in detail. The upcoming points are the issue which stands as problem or say as challenge before us.

3.1.1 Monitoring Keyboard & Retrieving the Scan Code



Keyboard is the basic input device, so far available for human to interact with computer and with any digital system. The case here that we have to manage to provide keys with each having special functioning. Then after, we have to keep an eye at any instance if any key is pressed, corresponding function should be done. So the task stands are:

- Determining Key states.
- Determining which key is pressed.
- Decide what kind of keypad should be used here.

3.1.2 Logic Circuit of Keyboard

This part is the main board, as an important part of the project too! This logic board shall be needed to monitor and read and send the data to computer containing the information about the key and its associated attributes like key states. The main question or say the challenge is that is any logical control unit is required for this or it can be done with simple digital logic components!?

3.1.3 Plugin Interface

What will be the interfacing type of the keyboard which we will construct? By this, we mean that what standards or what plug type we shall provide to get our keyboard connected to computer. While deciding this, we have to consider the current technologies and also not to forget about the earlier technology too!

3.1.4 Keyboard Fabrication

This is one of the challenges which got risks too, in terms of money, time and resources. By this what we means is that for fabrication of the keyboard we have to do either at production environment where the things are been carried out under professionals guidance or by self just like DIY (Do It Yourself).

4 Working & Methodology

4.1 Solutions

As we have defined and also recognised our problems which are determined in nature, now we shall analyse the problems and get to many solutions possible. The solutions are described in forth coming points and in the project the solutions are accepted considering various issues and points and reasons too.

4.2 Hardware

We will be going into detail of each and every parts which are used in this project and also into the software part which makes this embedded system into live.

4.2.1 Schematics & Layouts

4.2.1.1Push Button

A **push-button** (also spelled **pushbutton**) or simply **button** is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons (due to their physical nature) require a spring to return to their un-pushed state. Different people use different terms for the "pushing" of the button, such as **press**, **depress**, **mash**, and **punch**.

4.2.1.2 Crystals

The main terms used for crystal oscillators are explained as follows:

- **Operating temperature range:** Ambient temperature range within which the specified characteristics of a product can be achieved.
- **Frequency/temperature characteristics:**
  This means the upper and lower limits of the variation of output frequency when the ambient temperature is changed under
  rated conditions excluding the temperature. However, the amount of change in temperature is dependent on the normal temperature (25±2 °C).
- **Frequency/voltage coefficient:**
  This means the upper and lower limits of the variation of output frequency when the power supply voltage is changed under rated conditions excluding the power supply voltage. However, the temperature when the power supply voltage is changed is a normal temperature (25±2 °C) and the power supply voltage to be changed is a rated value.
- **Tristate (Standby function):**
  Any output circuit used for its output must be high impedance when output is OFF is installed.
- **Frequency adjustment range:**

Frequency variation from nominal frequency shows a minimum value under rated conditions (normal temperature, rated voltage, and rated load).

- **Frequency control characteristic:**
  This shows the variable frequency amount when voltage within a specified range is applied to the frequency control input. When voltage is applied, temperature is normal temperature (25±2 °C) and power supply voltage is a rated value.
- **Overall frequency tolerance:**
  This shows the frequency stability when the relationship with temperature, power supply, load, etc. is not taken into consideration.

4.2.1.2.1 Explanation of working of crystal oscillator

Crystal oscillators are generally classified into four types according to the difference in the general circuit. With the addition of a new type NDK now provides five types of crystal oscillator. Simple Packaged Crystal Oscillator (SPXO) An oscillator with no compensation or control of temperature. Generally, the AT-cut method is used for crystal units, and their Frequency-temperature characteristic shows a cubic curve. The frequency-temperature characteristic of a simple packaged crystal oscillator is almost the same as that of a crystal unit, and also shows as a cubic curve.

- Simple packaged crystal oscillators are mainly used as the reference oscillator of communication equipment, measuring instruments, controllers, etc. They are also used as the clock oscillator of computers, office automation equipment, information terminal devices, etc.
- Temperature-Compensated Crystal Oscillator (TCXO) Temperature-compensated crystal oscillators have temperature compensated circuits incorporated and are designed to have a satisfactory temperature characteristic across a wide temperature range. There are two types of temperature-compensated crystal oscillators:

one is combined with resistors and capacitors with thermistors used as thermo-sensitive devices, and the other is made up of temperature-compensated circuits with the devices in an LSI used as thermo-sensitive devices. A crystal oscillator of which the temperature-compensated circuits are integrated into an LSI normally has its oscillating circuits as a whole integrated into a single-chip LSI. Figure 1 shows an example of the temperature characteristics of a temperature-compensated crystal oscillator. TCXO has a good temperature characteristic and its power consumption is low. In addition, it is compact and light, with a short start time. Therefore, it is used as the reference oscillator of communication devices (mobile phones, GPS, land mobile radio systems, microwave communication and satellite communication systems) and of measuring instruments, such as frequency counters and synthesizers. A VC-TCXO (Voltage-Controlled TCXO) with an external supply voltage and the function of controlling frequencies (VC, AFC, (Automatic Frequency Control) is also treated as a TCXO.

4.2.1.3 Resistor

A **resistor** is a passive two-terminal electrical component that implements electrical resistance as a circuit element. The current through a resistor is in direct proportion to the voltage across the resistor's terminals. This relationship is represented by Ohm's law.

$$I = \frac{V}{R}$$

4.2.1.4 Capacitor

A **capacitor** (originally known as **condenser**) is a passive two-terminal electrical component used to store energy in an electric field. The forms of practical capacitors vary widely, but all contain at least two electrical conductors separated by a dielectric (insulator); for example, one common construction consists of metal foils separated by a thin layer of insulating film. Capacitors are widely used as parts of electrical circuits in many common electrical devices.

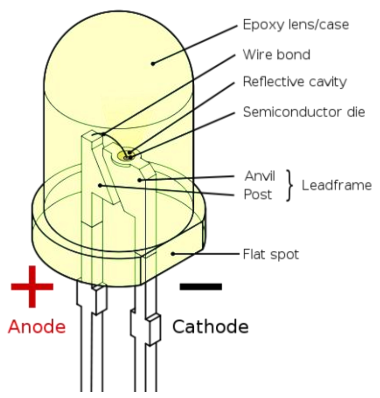
4.2.1.5 MCU (Microcontroller unit)

A **microcontroller** (sometimes abbreviated **µC**, **uC** or **MCU**) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications. Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

4.2.1.6 Printed Circuit Board

A **printed circuit board**, or **PCB**, is used to mechanically support and electrically connect electronic components using conductive pathways, tracks or signal traces etched from copper sheets laminated onto a non-conductive substrate. When the board has only copper tracks and features, and no circuit elements such as capacitors, resistors or active devices have been manufactured into the actual substrate of the board, it is more correctly referred to as **printed wiring board** (**PWB**) or **etched wiring board**. Use of the term **PWB** or **printed wiring board** although more accurate and distinct from what would be known as a true **printed circuit board**, has generally fallen by the wayside for many people as the distinction between **circuit** and **wiring** has become blurred. Today printed wiring (circuit) boards are used in virtually all but the simplest commercially produced electronic devices, and allow fully automated assembly processes that were not possible or practical in earlier era tag type circuit assembly processes. A PCB populated with electronic components is called a **printed circuit assembly** (**PCA**), **printed circuit board assembly** or **PCB Assembly** (**PCBA**). In informal use the term "PCB" is used both for bare and assembled boards, the context clarifying the meaning.

4.2.1.7 Light Emitting Diode



A **light-emitting diode** (**LED**) is a semiconductor light source. LEDs are used as indicator lamps in many devices and are increasingly used for other lighting. Appearing as practical electronic components in 1962, early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet, and infrared wavelengths, with very high brightness.

When a light-emitting diode is forward-biased (switched on), electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is often small in area (less than 1 mm^2), and integrated optical components may be used to shape its radiation pattern. LEDs present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. LEDs powerful enough for room lighting are relatively expensive and require more precise current and heat management than compact fluorescent lamp sources of comparable output.

4.2.1.8 Berg Pins

A **Berg connector** is a brand of electrical connector used in computer hardware. Berg connectors are manufactured by Berg Electronics Corporation of St. Louis, Missouri, a division of Framatome Connectors International. Berg connectors have a 2.54 mm (=.100 inch) pitch, pins are square (0.64 mm x 0.64 mm = approx. 0.025 x 0.025 inch), and usually come as single or double row connectors.

4.2.1.9 USB connectors

Universal Serial Bus (**USB**) is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication and power supply between computers and electronic devices. USB was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has become commonplace on other devices, such as smartphones, PDAs and video game consoles.

4.2.2 Circuitry Modules

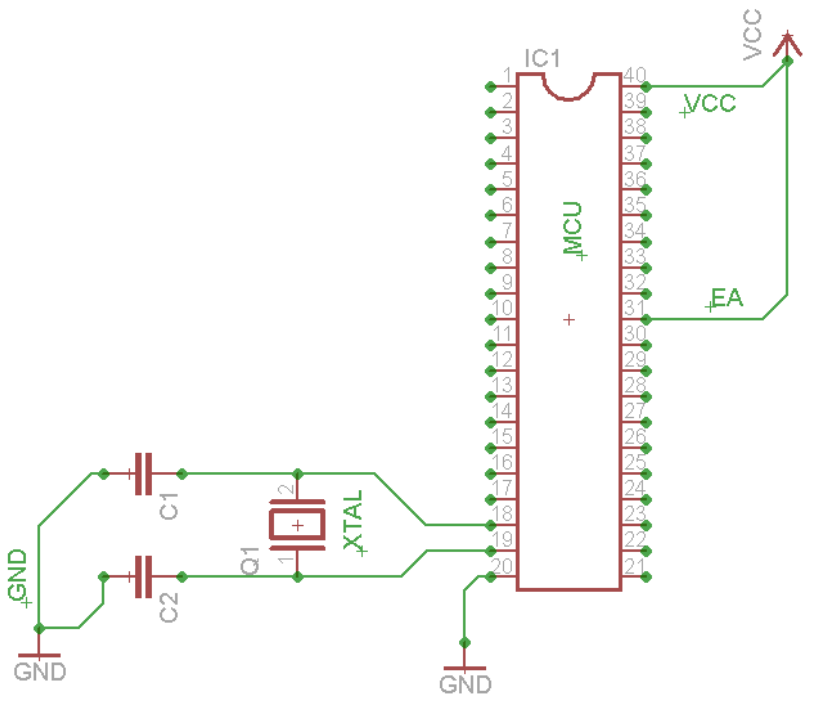
The hardware is been sub-divided into various interfacing modules. This approach will help us in assembling them with an ease. Also, if some kind of problems occurs, we will just have to work out over that particular module from where the problem is occurring. In forth coming sections you will find the schematics and also the layouts in later sections. Our Project in broader sense has been bifurcated:

1. Keyboard Module.
2. Controller Board Module.

4.2.2.1 Controller Board Module

Within this module, we will take a deep look in sub modules one by one. Here we will start from the MCU module…

4.2.2.1.1 Microcontroller Module

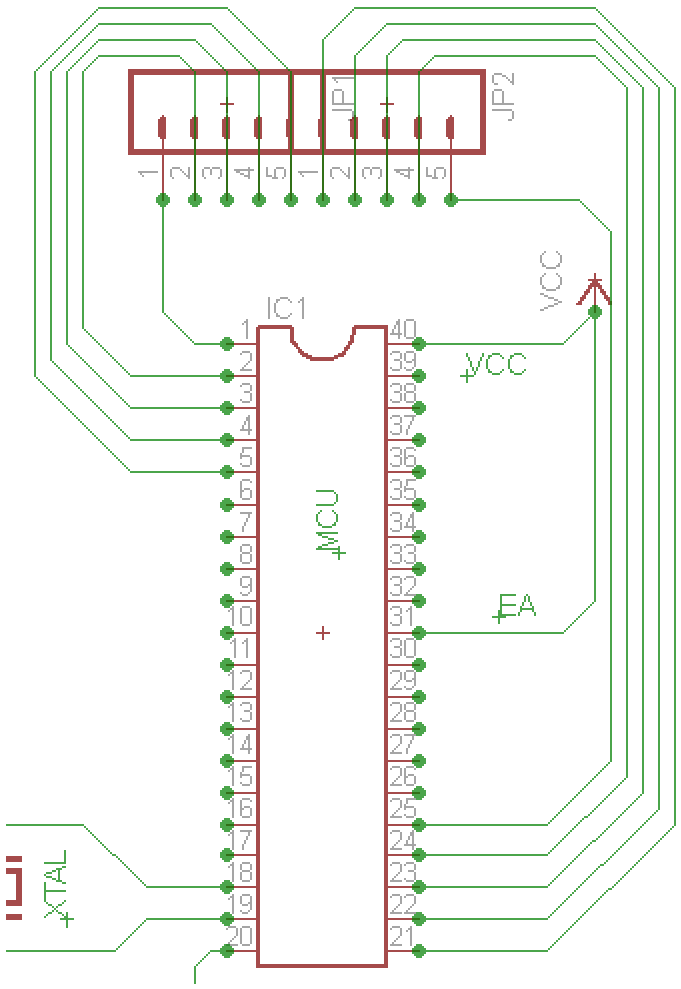


The figure which is shown above, is the basic bare bone circuit required for the 8051 MCU family to run and operate safely. For further information over this circuitry and other components which are connected to MCU, please refer to the datasheet of P89V51RD2FN/BN.

Components connected to MCU are:

- C1, C2 are capacitors, should be ceramic cap capacitors. Ranges from 12-22pF.
- XTAL- 0-40MHz is the range for this MCU. 11.0592MHz is used in our project.
- VCC- DC Current source voltage of 5V.
- GND- Ground signal.
- EA- To enable this MCU, according to the specification and directions mentioned in datasheet, we have to connect this pin to VCC for proper working.
- C1, C2 are capacitors which provided stability to oscillator and these values are recommended in the datasheet itself.
- Reset pin is been left and it will be connected to the USB to UART converter and that module will handle the Reset signal.

4.2.2.1.2 Connection of MCU with keyboard pins



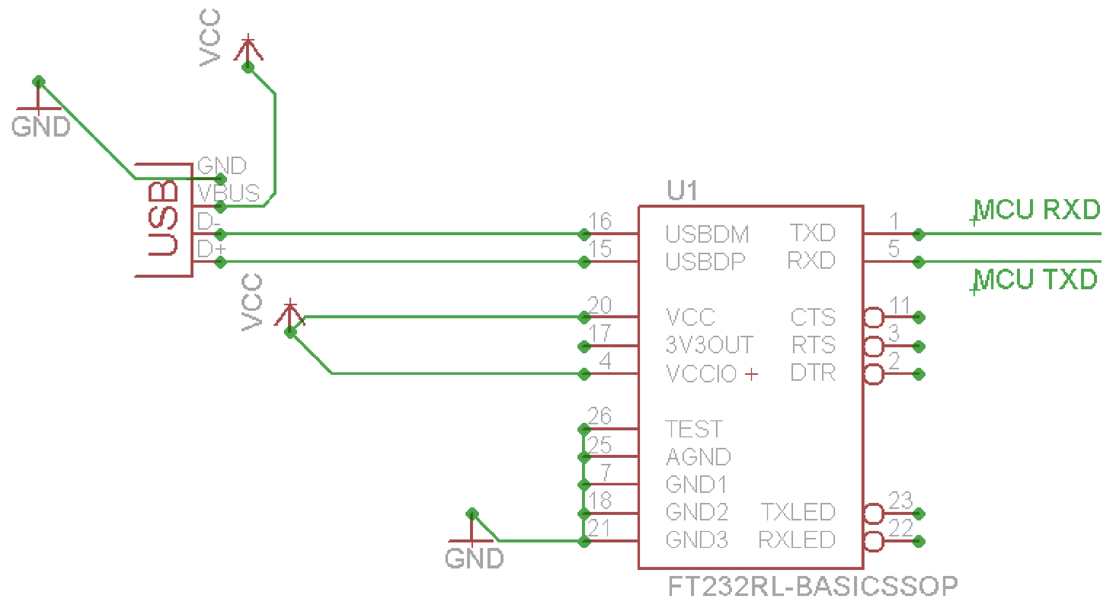
The above schematic is extended, and the keyboard pin connection with MCU is shown. Under this, a list is there which describes this above sketch.

Description:

- Keyboard: 5x5 sized, means 25 keys on board.
- This, 5 input lines and 5 output lines. Hence, there are 10 lines which would be connected to MCU.
- From left to right, first 5 bits, or say first 5 lines are connected to MCU's lower bit valued of Port 1.
- Rest of the five bits, or say the last five lines are connected to MCU's lower bit values of Port 2.
- JP1, JP2 are the jumpers which are actually berg pins (female) which will hold the keyboard into it.

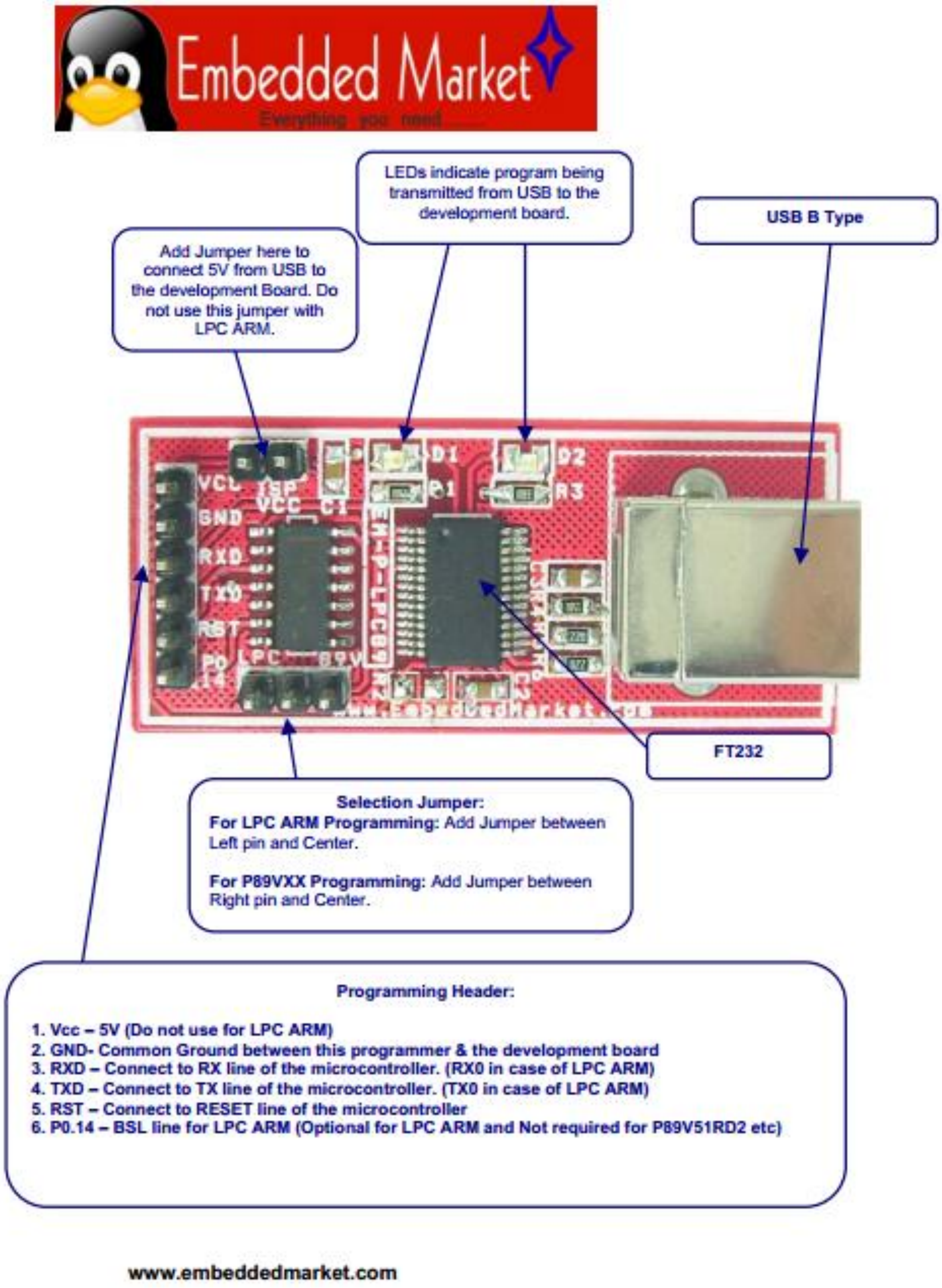
Next sub module coming is the USB interfacing module of this board. The power source, i.e. VCC comes from here only.

4.2.2.1.3USB interfacing



The above schematic is the sketch of the USB to UART serial bridge IC which is in SSOP-28 package. This IC provides us the direct communication from computer to MCU. As MCU works on 5V, and computer works at higher volts than this, thus this IC acts as an interface between PC to MCU. But to fabricate this IC at this level was very difficult and it was a matter of luck. Due to its tiny kind of size we have to pick a readily available soldered breakout module of this IC for proper working. In the left, the snap of that IC is shown. This IC is manufactured by FTDI Ltd. For further information regarding the same IC, please go through the data sheet of FT232RL.

The breakout module which is been used in our project for USB to TTL converter, all the information regarding the same is been mentioned under this.



LEDs indicate program being transmitted from USB to the development board.

Add Jumper here to connect 5V from USB to the development Board. Do not use this jumper with LPC ARM.

Selection Jumper:
For LPC ARM Programming: Add Jumper between Left pin and Center.

For P89VXX Programming: Add Jumper between Right pin and Center.

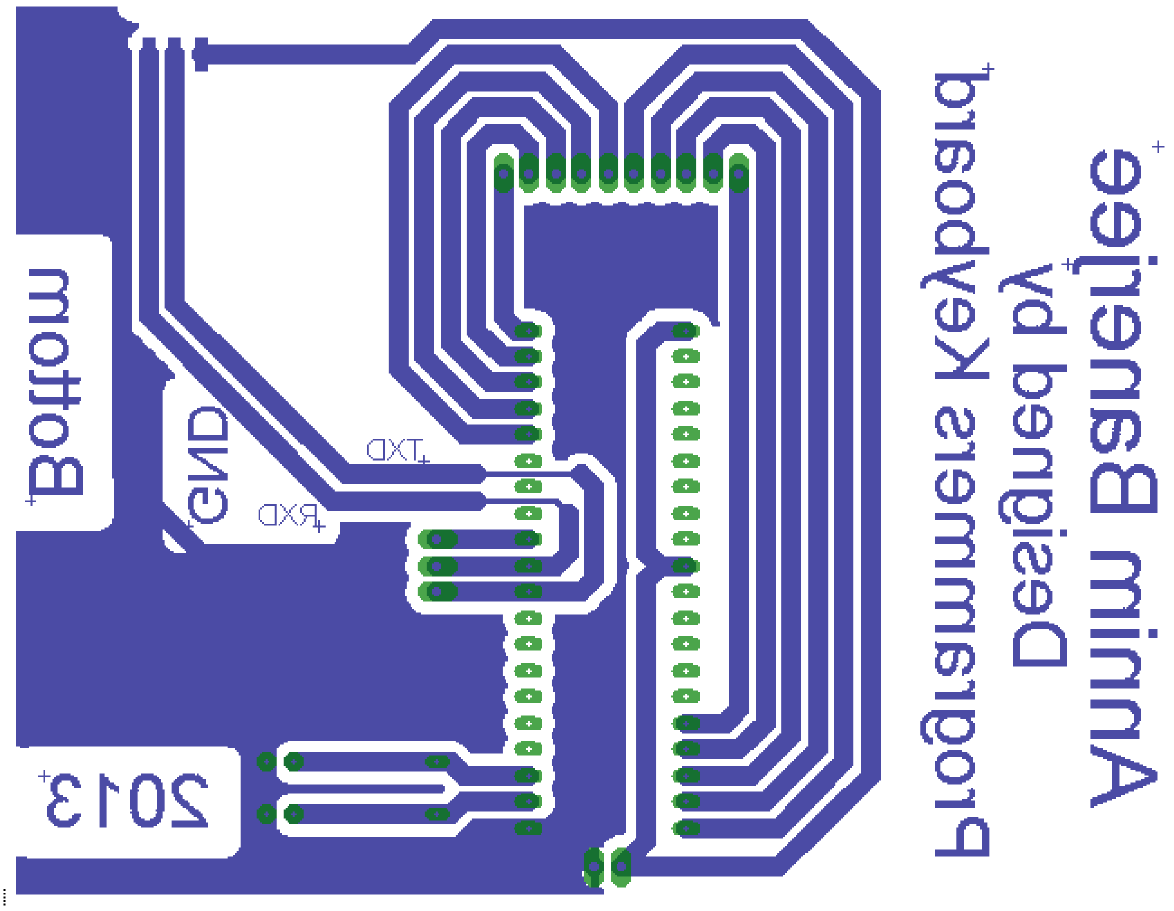
Programming Header:

1. Vcc – 5V (Do not use for LPC ARM)
2. GND- Common Ground between this programmer & the development board
3. RXD – Connect to RX line of the microcontroller. (RX0 in case of LPC ARM)
4. TXD – Connect to TX line of the microcontroller. (TX0 in case of LPC ARM)
5. RST – Connect to RESET line of the microcontroller
6. P0.14 – BSL line for LPC ARM (Optional for LPC ARM and Not required for P89V51RD2 etc)

www.embeddedmarket.com

The above datasheet snap describes about the USB to TTL converter which is been used in our project for communication purpose between PC to MCU. And the pin descriptions are also labelled in the diagram. VCC is coming from this location, i.e. from USB port of PC.

4.2.2.1.4 Controller Board Module Layout

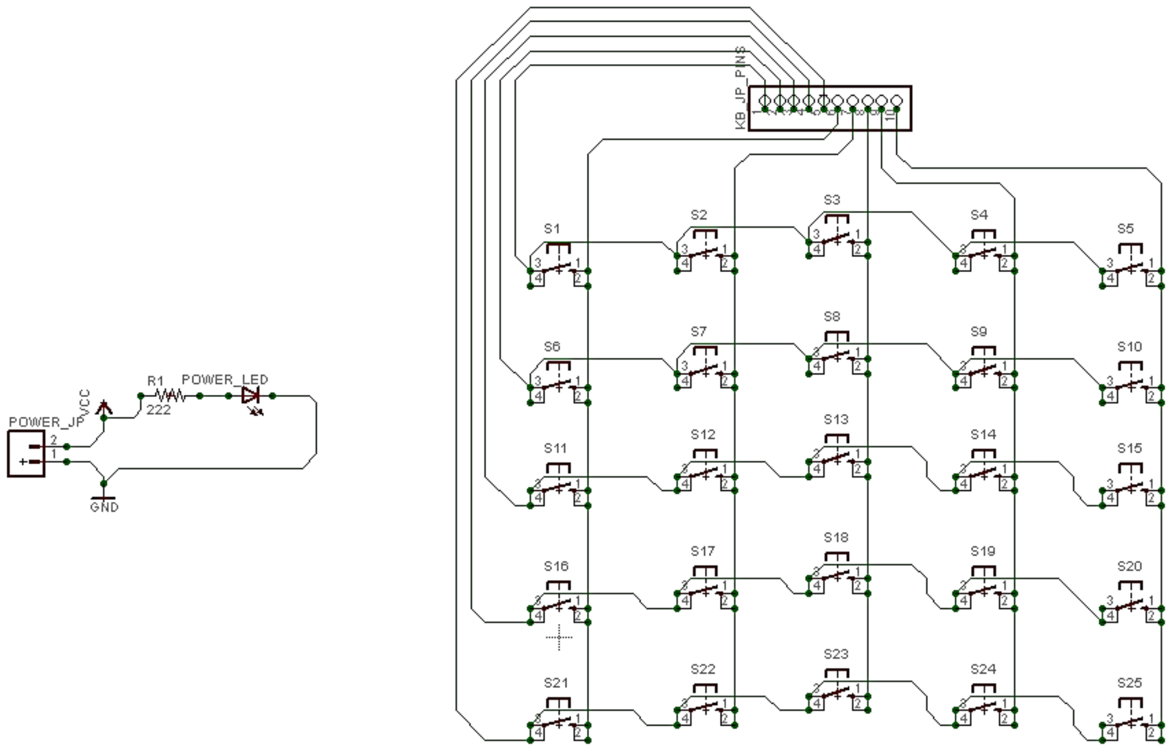


This is the circuit layout which will be used to fabricate the PCB board. This is the positive of the layout and already mirrored as this is the BOTTOM layer of the PCB. This layout is been created in Eagle CAD software.

4.2.2.2 Keyboard Module

Under this Module, we will be discussing the connection of the switches, aka Toggle switches and there arrangements on the board too.

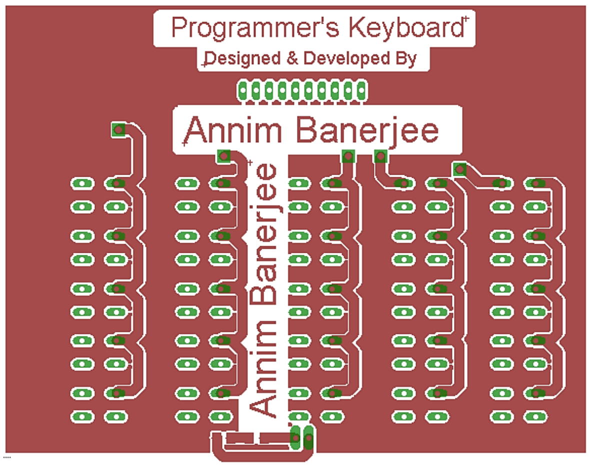
4.2.2.2.1 Matrix keyboard Schematic

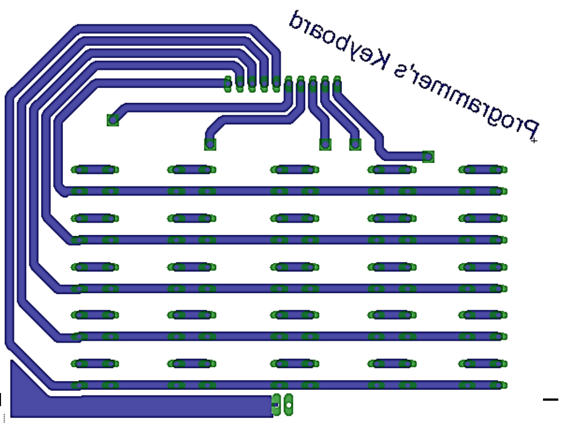


The above sketch is the matrix arrangement of the switches, aka push button with 4 pins. Similarly, the 10 pin berg pin (male) connectors are provided so the keyboard will stand over the controller board. These 10 lines are directly connected to MCU. And MCU is continually monitoring the signal values as HIGH or LOW.

4.2.2.2.2 Keyboard Layout

Our keyboard is of Double Layer PCB board and hence this has two layered layouts. This is also designed in same software, i.e. Eagle CAD software. (Left: Top layer, Right: Bottom Layer)



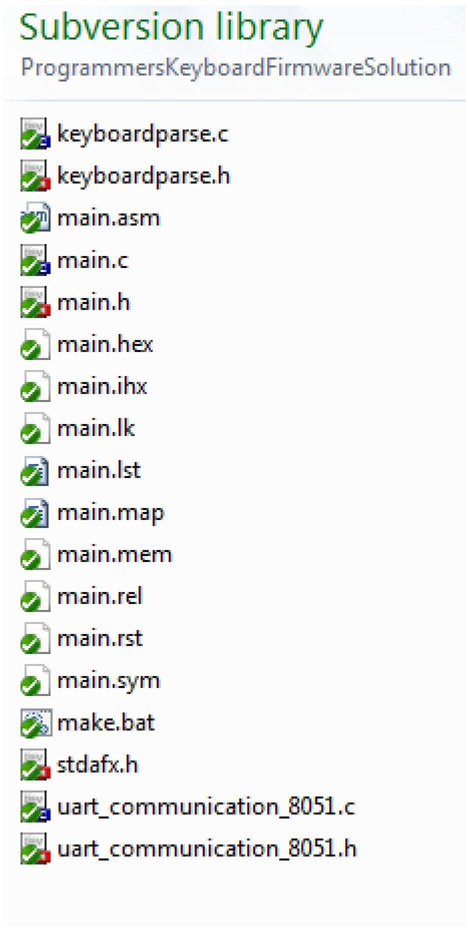


4.3 Software

Now the working of software comes into the picture. Now, we will be discussing over the firmware code which will be running in the MCU, whose soul task is to monitor the key's states and send the information of the pressed key to PC.

4.3.1 Firmware

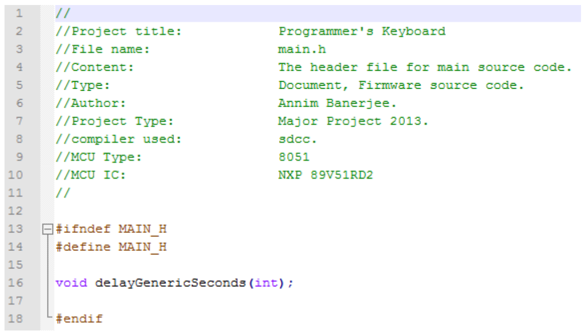
Taking about the firmware which has been written by us and has only one task is to send the information of the key's state to PC. The code is written in Embedded-C. Compiler used is SDCC (Small Device C Compiler). The Project file structure is shown below:



Subversion library
ProgrammersKeyboardFirmwareSolution

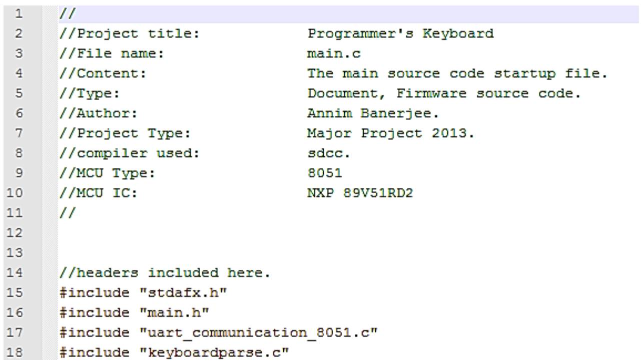
keyboardparse.c
keyboardparse.h
main.asm
main.c
main.h
main.hex
main.ihx
main.lk
main.lst
main.map
main.mem
main.rel
main.rst
main.sym
make.bat
stdafx.h
uart_communication_8051.c
uart_communication_8051.h

- uart_communication_8051.h: Header file for UART communication.
- Uart_communication_8051.c: Source file of the above declared prototypes.
- Keyboardparse.h: Header file for keyboard monitoring.
- Keyboardparse.c: Source file for the above declared prototypes.
- Main.h : A Header file.
- Main.c: The source file of main.h
- Stdafx.h: standard application header file which includes the necessary header files for 8051 MCU family.
- Main.hex: the hex file which is to be burned to MCU program memory. An intel hex file format.



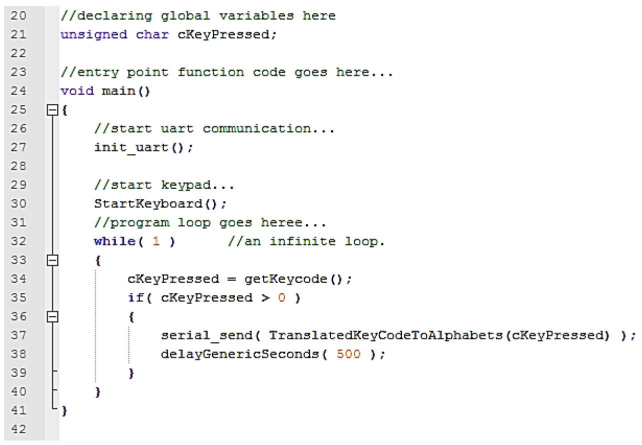
```
//
//Project title:            Programmer's Keyboard
//File name:                main.h
//Content:                  The header file for main source code.
//Type:                     Document, Firmware source code.
//Author:                   Annim Banerjee.
//Project Type:             Major Project 2013.
//compiler used:            sdcc.
//MCU Type:                 8051
//MCU IC:                   NXP 89V51RD2
//

#ifndef MAIN_H
#define MAIN_H

void delayGenericSeconds(int);

#endif
```

- **Main.h**
  Content of main.h is shown in left.



```
//
//Project title:            Programmer's Keyboard
//File name:                main.c
//Content:                  The main source code startup file.
//Type:                     Document, Firmware source code.
//Author:                   Annim Banerjee.
//Project Type:             Major Project 2013.
//compiler used:            sdcc.
//MCU Type:                 8051
//MCU IC:                   NXP 89V51RD2
//


//headers included here.
#include "stdafx.h"
#include "main.h"
#include "uart_communication_8051.c"
#include "keyboardparse.c"
```

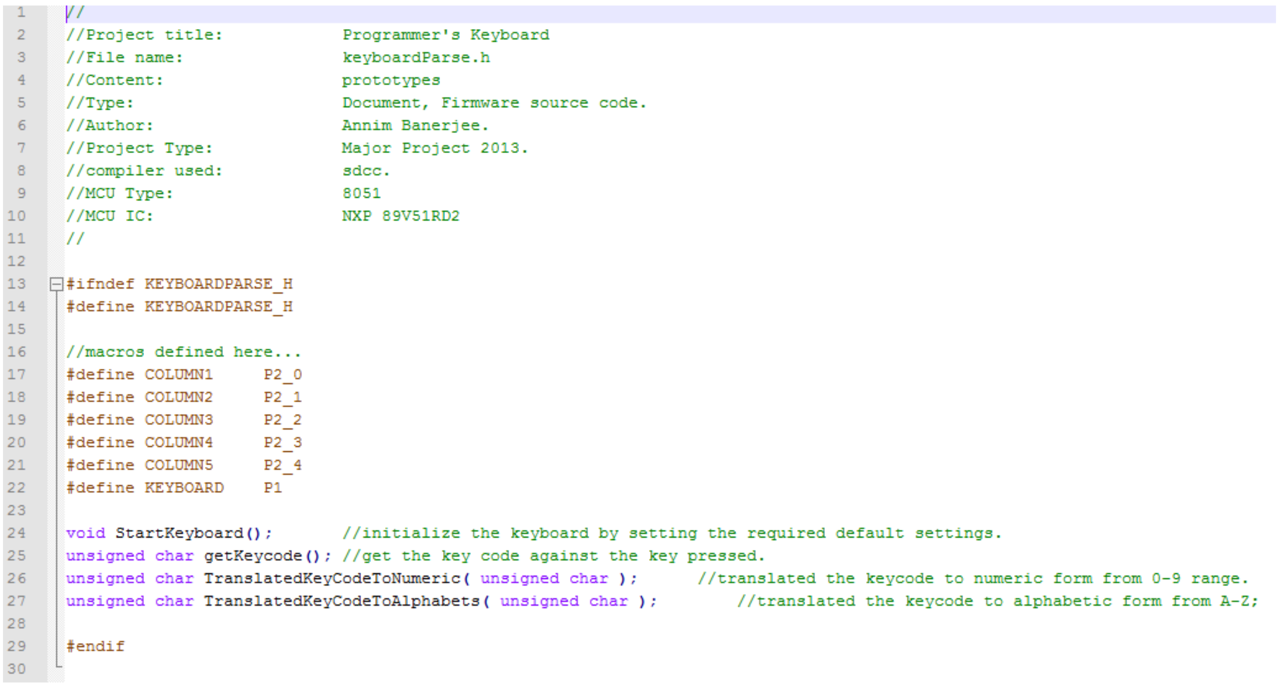
- **Main.c**



```
//declaring global variables here
unsigned char cKeyPressed;

//entry point function code goes here...
void main()
{
    //start uart communication...
    init_uart();

    //start keypad...
    StartKeyboard();
    //program loop goes heree...
    while( 1 )      //an infinite loop.
    {
        cKeyPressed = getKeycode();
        if( cKeyPressed > 0 )
        {
            serial_send( TranslatedKeyCodeToAlphabets(cKeyPressed) );
            delayGenericSeconds( 500 );
        }
    }
}
```

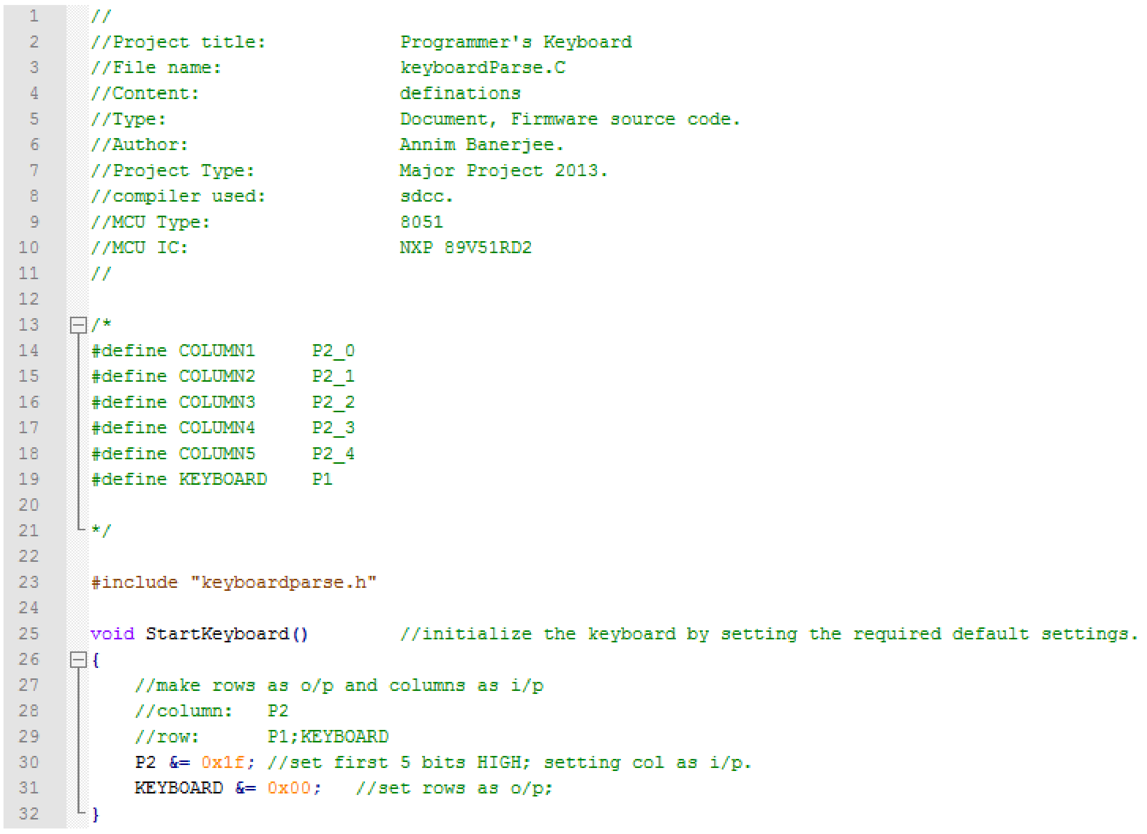
- Includes of this file shown in left.
- The entry point of the program is shown below.

- About the entry point:
  - There's a loop which continually get the key state.
  - That loop is an endless loop.
  - If key not pressed, do not send any information.
  - If key pressed, send information of the key pressed and which key is pressed.
- Before all, initiate the UART communication and also initiate the keyboard for its working.
- Keyboardparse.h
  - Picture shown below is the content of this header file. Few macros are defined and prototypes too are declared.



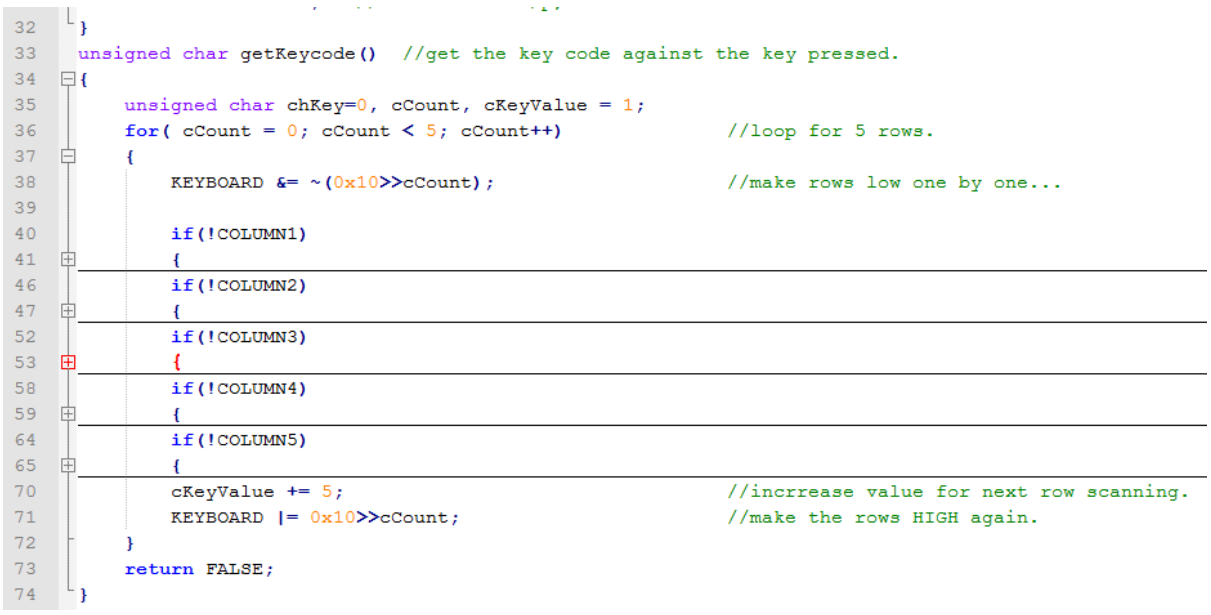
```
//
//Project title:            Programmer's Keyboard
//File name:                keyboardParse.h
//Content:                  prototypes
//Type:                     Document, Firmware source code.
//Author:                   Annim Banerjee.
//Project Type:             Major Project 2013.
//compiler used:            sdcc.
//MCU Type:                 8051
//MCU IC:                   NXP 89V51RD2
//

#ifndef KEYBOARDPARSE_H
#define KEYBOARDPARSE_H

//macros defined here...
#define COLUMN1     P2_0
#define COLUMN2     P2_1
#define COLUMN3     P2_2
#define COLUMN4     P2_3
#define COLUMN5     P2_4
#define KEYBOARD    P1

void StartKeyboard();       //initialize the keyboard by setting the required default settings.
unsigned char getKeycode(); //get the key code against the key pressed.
unsigned char TranslatedKeyCodeToNumeric( unsigned char );      //translated the keycode to numeric form from 0-9 range.
unsigned char TranslatedKeyCodeToAlphabets( unsigned char );        //translated the keycode to alphabetic form from A-Z;

#endif
```

- Keyboardparse.c



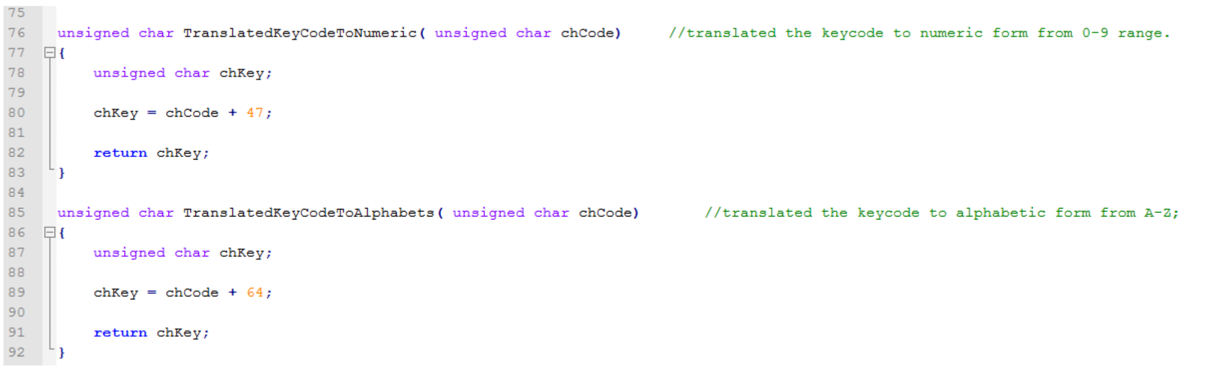
```
//
//Project title:            Programmer's Keyboard
//File name:                keyboardParse.C
//Content:                  definations
//Type:                     Document, Firmware source code.
//Author:                   Annim Banerjee.
//Project Type:             Major Project 2013.
//compiler used:            sdcc.
//MCU Type:                 8051
//MCU IC:                   NXP 89V51RD2
//

/*
#define COLUMN1     P2_0
#define COLUMN2     P2_1
#define COLUMN3     P2_2
#define COLUMN4     P2_3
#define COLUMN5     P2_4
#define KEYBOARD    P1

*/

#include "keyboardparse.h"

void StartKeyboard()        //initialize the keyboard by setting the required default settings.
{
    //make rows as o/p and columns as i/p
    //column:   P2
    //row:      P1;KEYBOARD
    P2 &= 0x1f; //set first 5 bits HIGH; setting col as i/p.
    KEYBOARD &= 0x00;   //set rows as o/p;
}
```

The above shown code snap showing the definition of the *StartKeyboard()* method. The comments are there which describes the code itself.



```
}
unsigned char getKeycode()  //get the key code against the key pressed.
{
    unsigned char chKey=0, cCount, cKeyValue = 1;
    for( cCount = 0; cCount < 5; cCount++)            //loop for 5 rows.
    {
        KEYBOARD &= ~(0x10>>cCount);                  //make rows low one by one...

        if(!COLUMN1)
        {
        if(!COLUMN2)
        {
        if(!COLUMN3)
        {
        if(!COLUMN4)
        {
        if(!COLUMN5)
        {
        cKeyValue += 5;                               //incrrease value for next row scanning.
        KEYBOARD |= 0x10>>cCount;                     //make the rows HIGH again.
    }
    return FALSE;
}
```

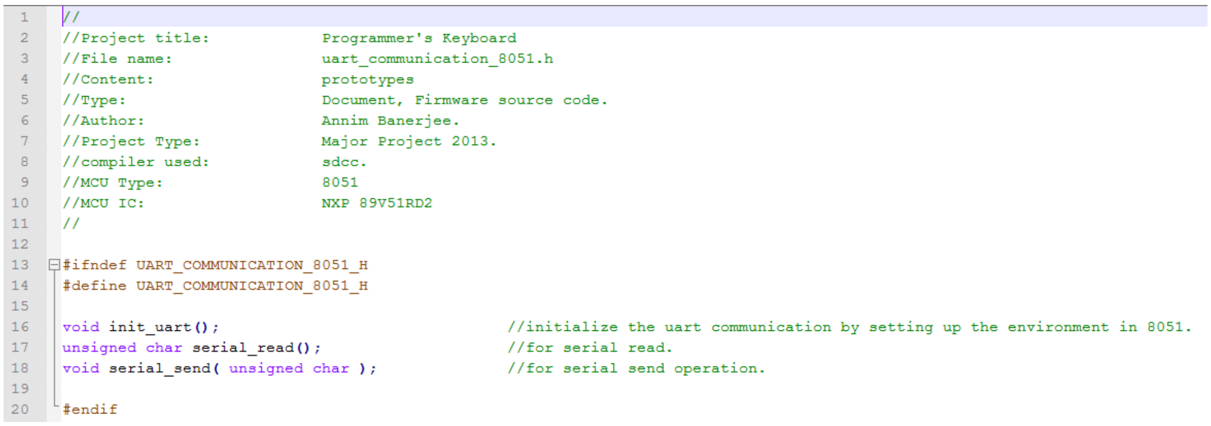
The above shown code snap is the code where we keep looking for the key states and if any of the key get pressed, immediately a corresponding value is generated. The comments are there which gives a pretty clear description about the code.



```
unsigned char TranslatedKeyCodeToNumeric( unsigned char chCode)     //translated the keycode to numeric form from 0-9 range.
{
    unsigned char chKey;

    chKey = chCode + 47;

    return chKey;
}

unsigned char TranslatedKeyCodeToAlphabets( unsigned char chCode)       //translated the keycode to alphabetic form from A-Z;
{
    unsigned char chKey;

    chKey = chCode + 64;

    return chKey;
}
```

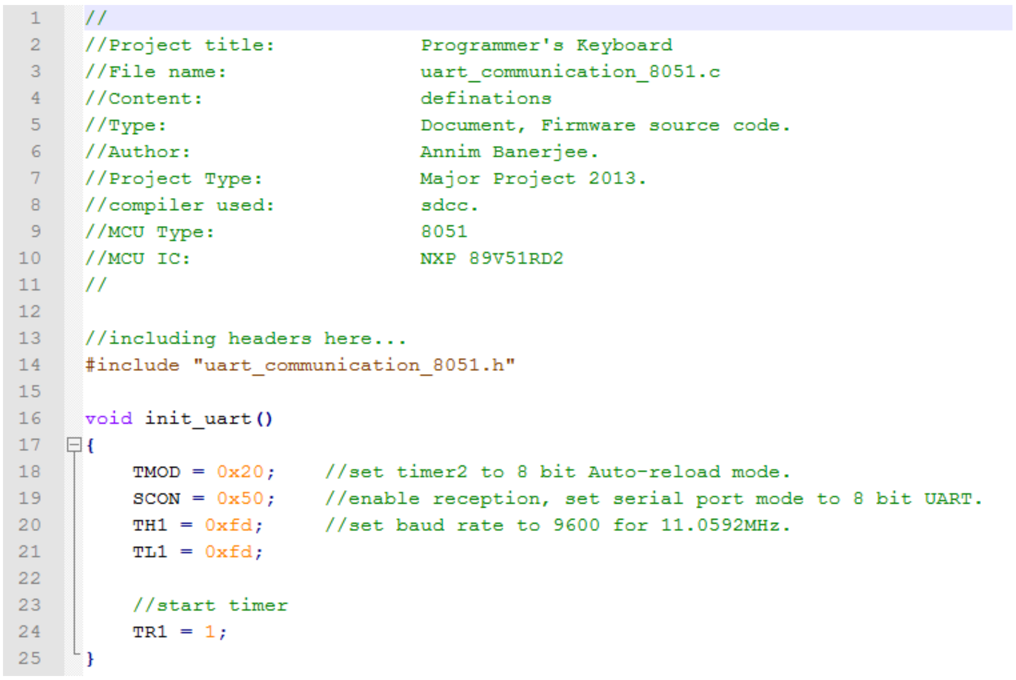
These are two functions which convert the alphabet to number and from number to alphabet. One of the function is been called in the main entry point and the translated value is been send to the PC end for further processing.

➢ UART communication header content



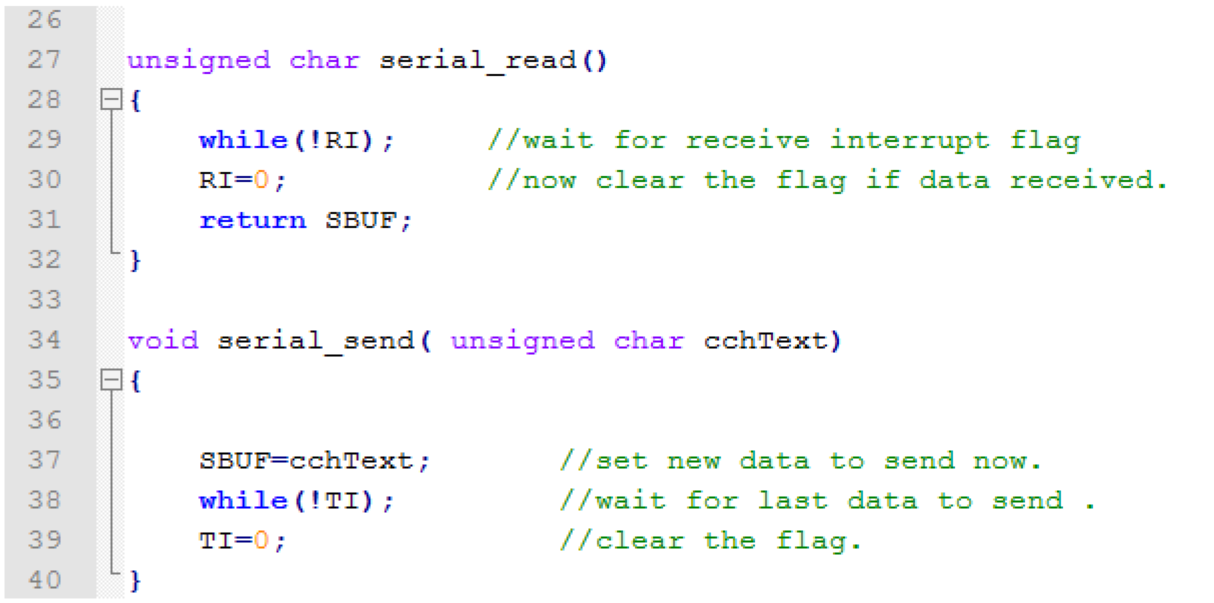
```
//
//Project title:            Programmer's Keyboard
//File name:                uart_communication_8051.h
//Content:                  prototypes
//Type:                     Document, Firmware source code.
//Author:                   Annim Banerjee.
//Project Type:             Major Project 2013.
//compiler used:            sdcc.
//MCU Type:                 8051
//MCU IC:                   NXP 89V51RD2
//

#ifndef UART_COMMUNICATION_8051_H
#define UART_COMMUNICATION_8051_H

void init_uart();                                //initialize the uart communication by setting up the environment in 8051.
unsigned char serial_read();                     //for serial read.
void serial_send( unsigned char );               //for serial send operation.

#endif
```

➢ UART communication source file content, with comments too.



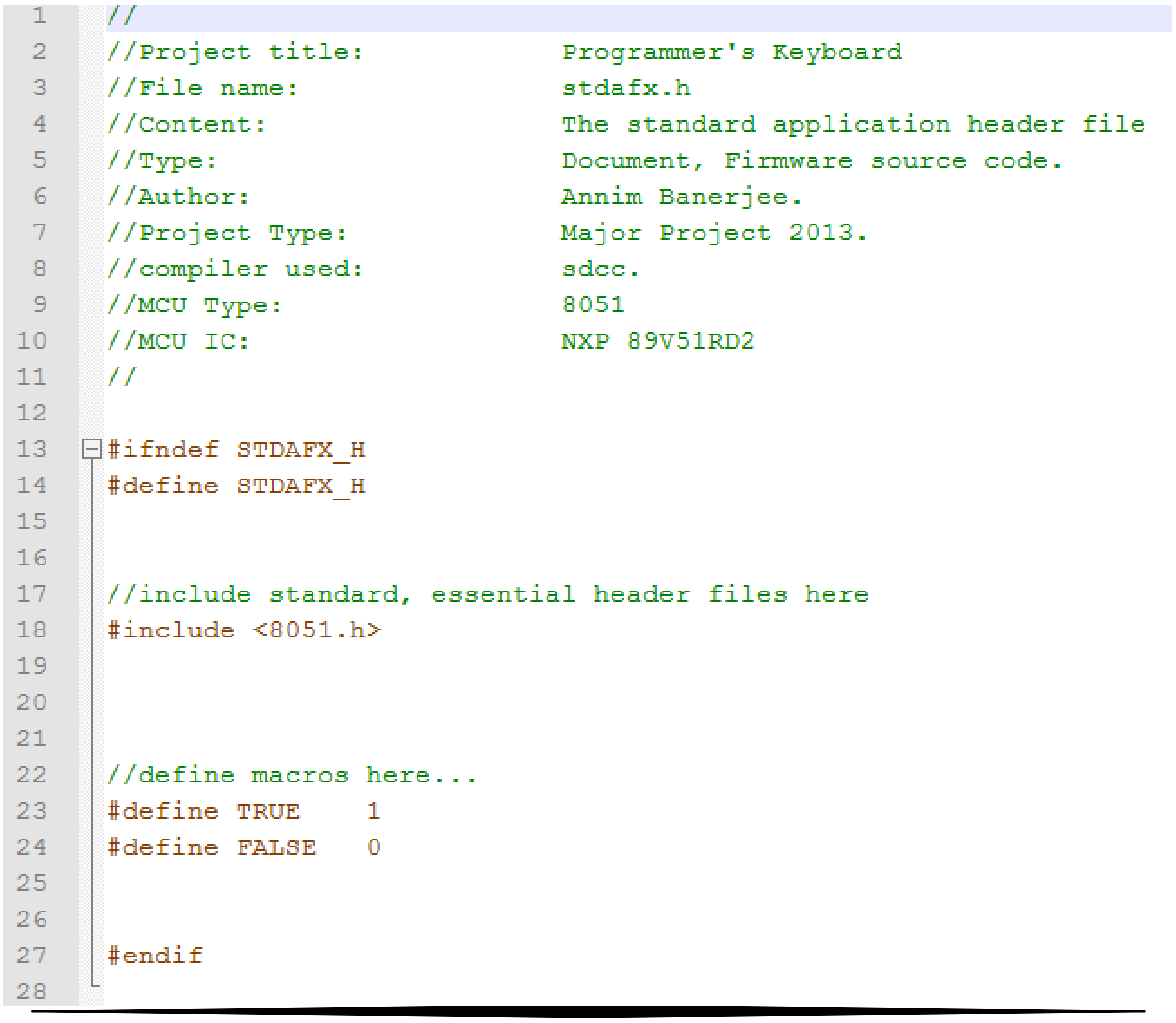
```
//
//Project title:            Programmer's Keyboard
//File name:                uart_communication_8051.c
//Content:                  definations
//Type:                     Document, Firmware source code.
//Author:                   Annim Banerjee.
//Project Type:             Major Project 2013.
//compiler used:            sdcc.
//MCU Type:                 8051
//MCU IC:                   NXP 89V51RD2
//

//including headers here...
#include "uart_communication_8051.h"

void init_uart()
{
    TMOD = 0x20;    //set timer2 to 8 bit Auto-reload mode.
    SCON = 0x50;    //enable reception, set serial port mode to 8 bit UART.
    TH1 = 0xfd;     //set baud rate to 9600 for 11.0592MHz.
    TL1 = 0xfd;

    //start timer
    TR1 = 1;
}
```

- UART Read and Write function content with comments



```

unsigned char serial_read()
{
    while(!RI);     //wait for receive interrupt flag
    RI=0;           //now clear the flag if data received.
    return SBUF;
}

void serial_send( unsigned char cchText)
{

    SBUF=cchText;       //set new data to send now.
    while(!TI);         //wait for last data to send .
    TI=0;               //clear the flag.
}
```

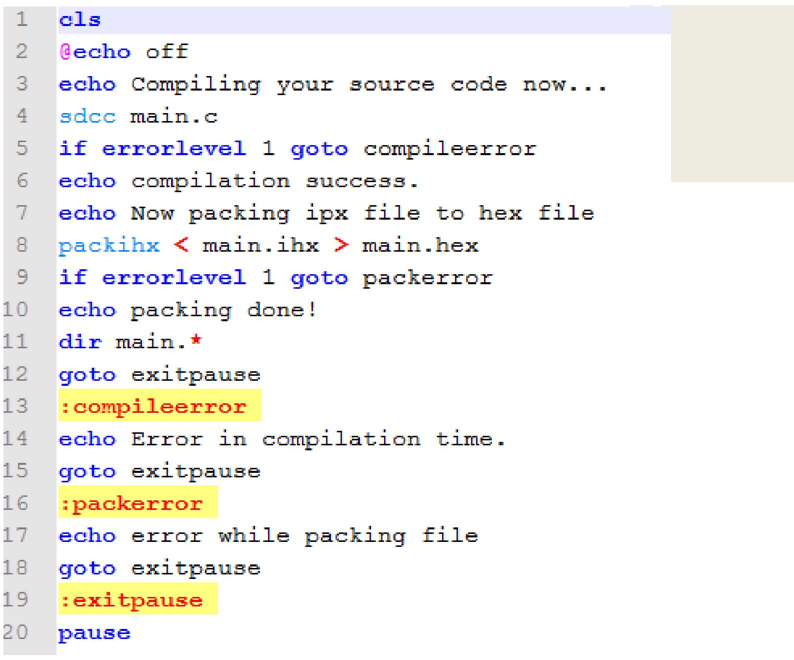
- Stdafx.h: Contents with comments



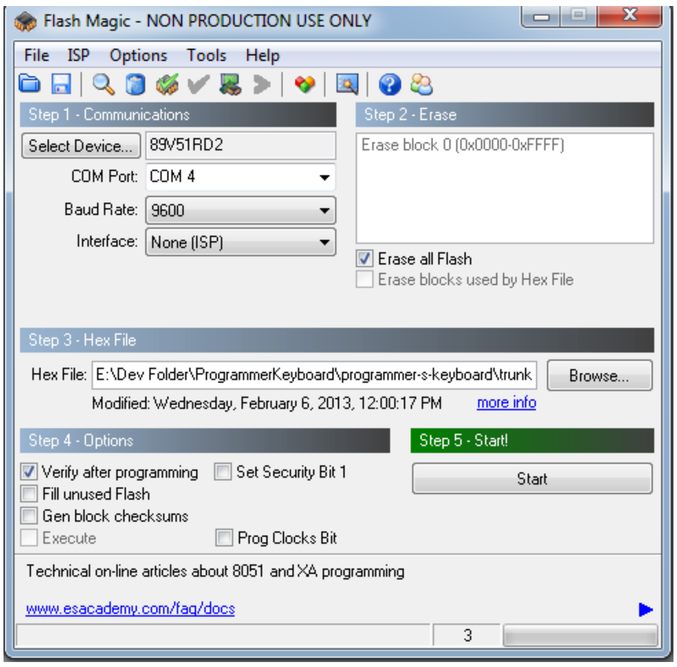
```
//
//Project title:              Programmer's Keyboard
//File name:                  stdafx.h
//Content:                    The standard application header file
//Type:                       Document, Firmware source code.
//Author:                     Annim Banerjee.
//Project Type:               Major Project 2013.
//compiler used:              sdcc.
//MCU Type:                   8051
//MCU IC:                     NXP 89V51RD2
//

#ifndef STDAFX_H
#define STDAFX_H


//include standard, essential header files here
#include <8051.h>



//define macros here...
#define TRUE    1
#define FALSE   0


#endif

```

- Makefile.bat

This file is been created for compiling the source code main.c and the compiler in result, produces the hex file. That hex file is then burnt on to MCU using *FlahsMagic* software provided by NXP Ltd.



```
cls
@echo off
echo Compiling your source code now...
sdcc main.c
if errorlevel 1 goto compileerror
echo compilation success.
echo Now packing ipx file to hex file
packihx < main.ihx > main.hex
if errorlevel 1 goto packerror
echo packing done!
dir main.*
goto exitpause
:compileerror
echo Error in compilation time.
goto exitpause
:packerror
echo error while packing file
goto exitpause
:exitpause
pause
```



Flash Magic - NON PRODUCTION USE ONLY

Step 1 - Communications
Select Device... 89V51RD2
COM Port: COM 4
Baud Rate: 9600
Interface: None (ISP)

Step 2 - Erase
Erase block 0 (0x0000-0xFFFF)
[✓] Erase all Flash
[ ] Erase blocks used by Hex File

Step 3 - Hex File
Hex File: E:\Dev Folder\ProgrammerKeyboard\programmer-s-keyboard\trunk
Modified: Wednesday, February 6, 2013, 12:00:17 PM

Step 4 - Options
[✓] Verify after programming
[ ] Set Security Bit 1
[ ] Fill unused Flash
[ ] Gen block checksums
[ ] Execute
[ ] Prog Clocks Bit

Technical on-line articles about 8051 and XA programming
www.esacademy.com/faq/docs

This window is of *FlashMagic* software via which the hex file is been dumped in to MCU.

On the right side shown device is the programmer used to port the hex file in to MCU and this is also been used for serial communication from MCU to PC. This device is been provided by Embedded Market.

4.3.2 Software on PC end

4.3.2.1 Abstract

"Programmer's Keyboard", named software is the software which will be sitting in PC front and it will be monitoring the keyboard's information as it comes up to the port. The purpose of this software is to read the data coming on com port and fetch up the key so pressed over the current working window.

This software is been developed in two different programming languages, i.e. in C++ and in C#. Aim after developing the same in two different languages was to gain experience and also see the prototyping experiences which one can see and observe the difference in levels of quality and dependency. In the next few pages, a detailed description of *"Programmer's Keyboard"* is been mentioned regarding the modules, coding and also about UI.

4.3.3 Requirements & Analysis

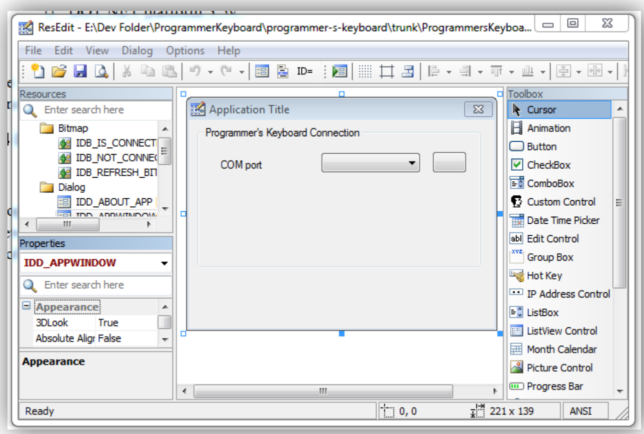
Gathering the requirement, which will be useful to design, develop and frame this software at its best. Below this, many point are mentioned which are then and there analysed too!

- What you need for establishing communication over a COM port?
  - The COM port name.
  - Baud Rate.
  - Parity.
  - Stop Bit.
  - IO writing APIs.
  - The Device Drivers.
- At which languages this keyboard will work for?
  - C programming language.
  - C++ programming language.
  - Java programming language.
  - C# programming language.
  - VB programming language.
- To connect to the device from UI, how?
  - A button will be enough to provide a way to interact for connecting to device and also to disconnect, as the COM port will close on disconnecting from *programmer's keyboard.*
- Controlling point on the go…
  - There should be some kind a way to change the currently working programming language while the application is running.
  - Application should be minimized or shouldn't appear in taskbar.
  - Hence, after minimization, the application should go in notification tray with the notifying icon form.
- Developing Technologies and Tools
  - Native development.

- Visual C++ 2010 Express edition IDE.
- Res Edit Resource editor.
- For DOT.NET platform, Visual C# 2010 Express edition.
- DOT.NET platform 3.5v.

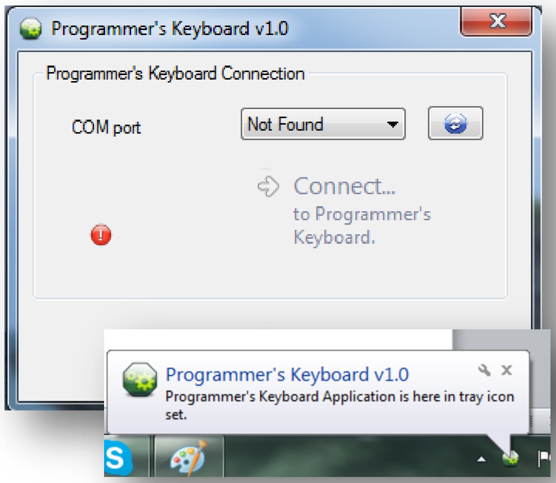
In the next segment, a detailed discussion over how these requirements are been then designed and implemented.

4.3.4 Designing



The designing of UI, especially under windows platform is very easy now a day. But, developing the UI in native coding style is bit tedious job, not taking support of any kind of framework. Thus, to do this, we have to use ResEdit resource editor, with the help of this IDE kind of software; we can portray the UI of our application. This snap to the left, is the ResEdit Resource editor IDE, and the window show is our applications main window which will come up above all for connecting and disconnecting from our keyboard device. Once the resource script is been generated by this IDE, we can import the script to our VS2010 VC++ IDE which will be reflected in solution explorer as show to left, and compile the program to generate window.

In DOT.NET platform, the UI designing is very fluid and convenient as there will be provided with many rich controls and also with drag-n-drop feature to design UI. For now, continuing with the native development, further more designing of UI is been done by hard coding too. This is been done by using the windows sdk, in which APIs are been provided to do such task either on runtime or at the initial phase of window creation. Snap to the left, is a dummy window created by using resource script and also few hard codes.


Programmer's Keyboard v1.0
Programmer's Keyboard Connection
COM port
Not Found
Connect...
to Programmer's Keyboard.
Programmer's Keyboard v1.0
Programmer's Keyboard Application is here in tray icon set.

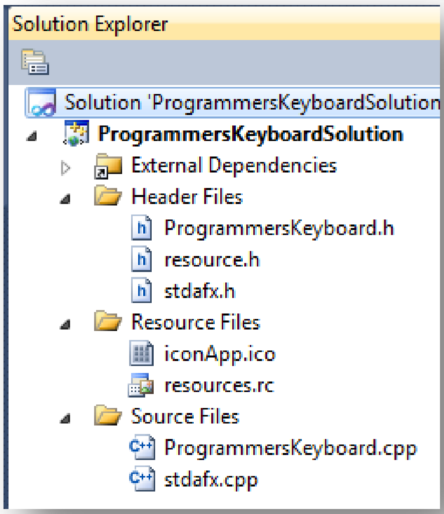
Programmer's keyboard application will be minimized to let the user work over, thus, there is a provision to put the application in notification tray. This feature is there in both the versions.

4.3.5 Implementation

In this segment, after having done with designing, we will not discussing the coding portion, especially regarding developing the application under C programming language, i.e. native development.

- Considerations
    - Codes are quite similar in two versions.
    - UI is almost identical.

That's why, the explanation of the application, and how it's been implemented, is mentioned for once and goes with both the versions with little changes.

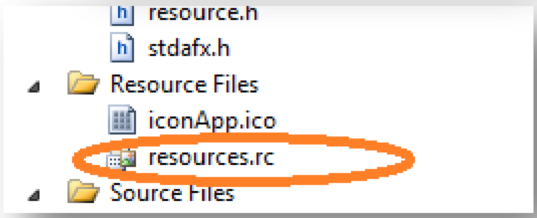


```
Solution Explorer
Solution 'ProgrammersKeyboardSolution
  ProgrammersKeyboardSolution
    External Dependencies
    Header Files
      ProgrammersKeyboard.h
      resource.h
      stdafx.h
    Resource Files
      iconApp.ico
      resources.rc
    Source Files
      ProgrammersKeyboard.cpp
      stdafx.cpp
```

4.3.5.1 Solution File Structure

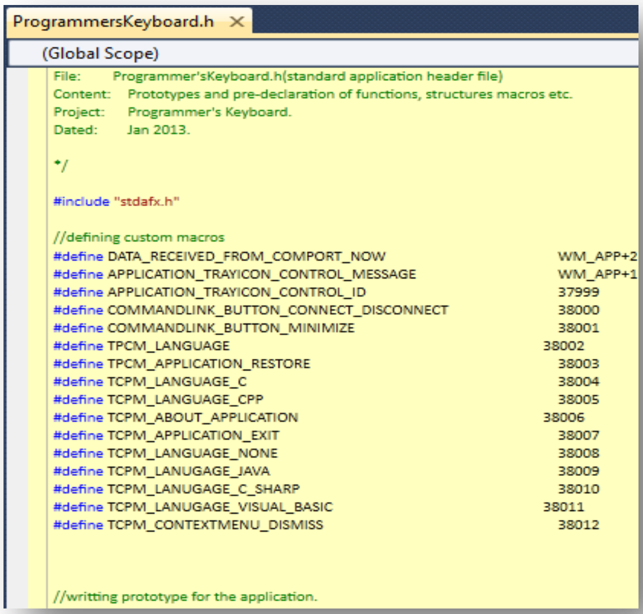
According to the Solution file structure so formed in project, according to that sequence we shall discuss.

- ProgrammersKeyboard.h: The header file having all the prototypes declared, like user defined functions and procedures, some global variables and MACROS too.
- resource.h: a header file specially created for having the MACROS as ID for the resources.
- stdafx.h: stands for "standard application framework", this file contains the essential header files included for GUI programming in window platform.



```
resource.h
stdafx.h
Resource Files
  iconApp.ico
  resources.rc
Source Files
```

-
- iconApp.ico: the applications logo/icon as resource.
- resource.rc: the only resource script file so generated from ResEdit resource editor IDE.
- ProgrammersKeyboard.cpp: The source file corresponding toProgrammersKeyboard.h header file, which contains the defination of all the functions and procedures like windowing and other event handlers too.



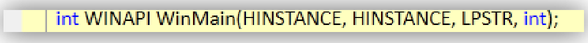
```
ProgrammersKeyboard.h
(Global Scope)
File:       Programmer'sKeyboard.h(standard application header file)
Content:    Prototypes and pre-declaration of functions, structures macros etc.
Project:    Programmer's Keyboard.
Dated:      Jan 2013.

*/

#include "stdafx.h"

//defining custom macros
#define DATA_RECEIVED_FROM_COMPORT_NOW                    WM_APP+2
#define APPLICATION_TRAYICON_CONTROL_MESSAGE              WM_APP+1
#define APPLICATION_TRAYICON_CONTROL_ID                   37999
#define COMMANDLINK_BUTTON_CONNECT_DISCONNECT             38000
#define COMMANDLINK_BUTTON_MINIMIZE                       38001
#define TPCM_LANGUAGE                                   38002
#define TPCM_APPLICATION_RESTORE                          38003
#define TCPM_LANGUAGE_C                                   38004
#define TCPM_LANGUAGE_CPP                                 38005
#define TCPM_ABOUT_APPLICATION                          38006
#define TCPM_APPLICATION_EXIT                             38007
#define TCPM_LANGUAGE_NONE                                38008
#define TCPM_LANUGAGE_JAVA                                38009
#define TCPM_LANUGAGE_C_SHARP                             38010
#define TCPM_LANUGAGE_VISUAL_BASIC                      38011
#define TCPM_CONTEXTMENU_DISMISS                          38012



//writting prototype for the application.
```

- stdafx.cpp: the source file corresponding to the stdafx.h header file.

Contents explanation of each file is been discussed in next segment.

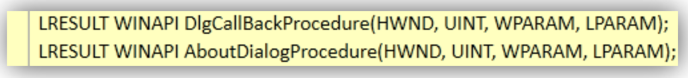
4.3.5.1.1 ProgrammersKeyboard.h

The header file having all the prototypes declared, like user defined functions and procedures, some global variables and MACROS too. There is a list mentioned below gives you the description of each and every user defined function as decalred in this header file.



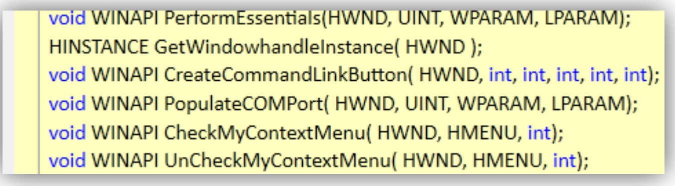
```
int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int);
```

- WinMain function is not a user defined function; this is actually the **entry point** of the GUI based program. It always takes four parameters which are the new instance data type which OS provides to application when launched, next argument is also instance handle which represent any previous instance if running in memory already, third argument is command line argument and the fourth one is the option as integer value which specifies the window state at the time of launching the window. This function also returns an integer type data to OS after the application ends up, which is as an exit code.



```
LRESULT WINAPI DlgCallBackProcedure(HWND, UINT, WPARAM, LPARAM);
LRESULT WINAPI AboutDialogProcedure(HWND, UINT, WPARAM, LPARAM);
```

- These two functions shown on to right are the prototypes of the CallBack procedures where all the event handlers corresponding to the window are been written in. These CallBack procedures takes four arguments and this is also not a user defined function, these format or say the function prototype is been designed and said mandatory by Microsoft itself and for further information regarding to these functions, please refer Microsoft APIs documentation. These functions take HWND as handle to window as first parameter, UINT i.e. unsigned integer as second argument which specifies the Window Message or say the intimation of the event, third and fourth one are the Wide Parameter and Low parameter which comes with the windows message giving information as companion to it. WPARAM and LPARAM are 32 bit long data contains extra information many a times.



```
void WINAPI PerformEssentials(HWND, UINT, WPARAM, LPARAM);
HINSTANCE GetWindowhandleInstance( HWND );
void WINAPI CreateCommandLinkButton( HWND, int, int, int, int, int);
void WINAPI PopulateCOMPort( HWND, UINT, WPARAM, LPARAM);
void WINAPI CheckMyContextMenu( HWND, HMENU, int);
void WINAPI UnCheckMyContextMenu( HWND, HMENU, int);
```

- Rest of the functions are user defined functions. *PerformEssentials* takes four parameters and by using that it executes few mandatory steps to put the window in front on screen with all basic settings. Like setting up the icon, title text, finding com port and putting it in combo box list as data and setting the buttons states for connecting and disconnecting, and putting image on button and other supplements task.

- *PopulateCOMPort* is another user defined function which takes same four arguments and fill up the combo box with the all available COM port on to the system. This function is thus used for refreshing purpose too!!!

- Other functions so named in such a way that the name itself can describe its purpose, what it is for!

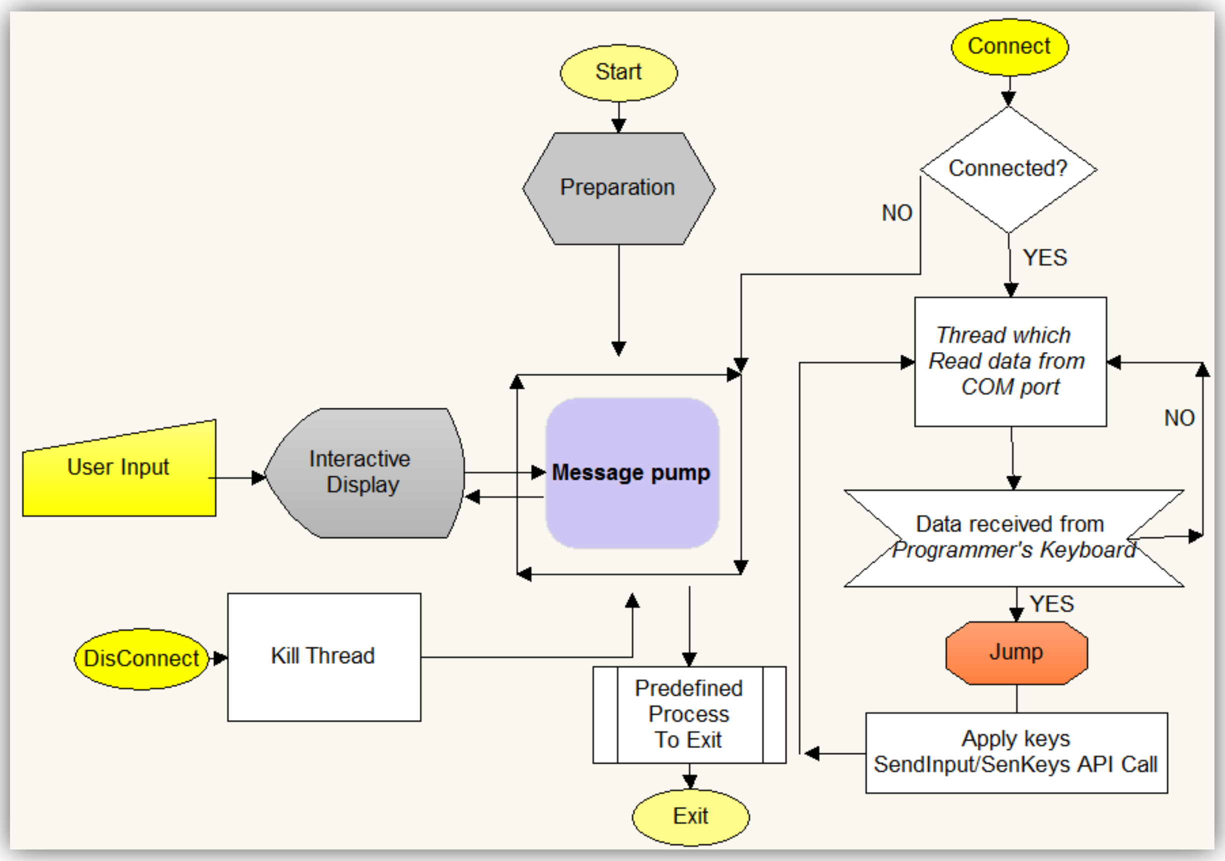
```
DWORD WINAPI  ReadExistingFromComPort( void * );
```

- This is also a user defined function which has only one job is to keep seeking for any data on COM port and get that data to some kind of storage area.

```
DWORD WINAPI PressShiftKey();
DWORD WINAPI ReleaseShiftKey();
DWORD WINAPI HitEnterKey();
DWORD WINAPI HitTabKey();
DWORD WINAPI HitSpaceKey();
DWORD WINAPI hitBackSpace();
```

- *PressShiftKey*, *ReleaseShiftKey*, *HitEnterKey*, *HitTabKey*, *HitSpaceKey* are some user defined functions as shown on right, which simulates the event as key is pressed on immediate keyboard.

4.3.5.2 Control Flow of software

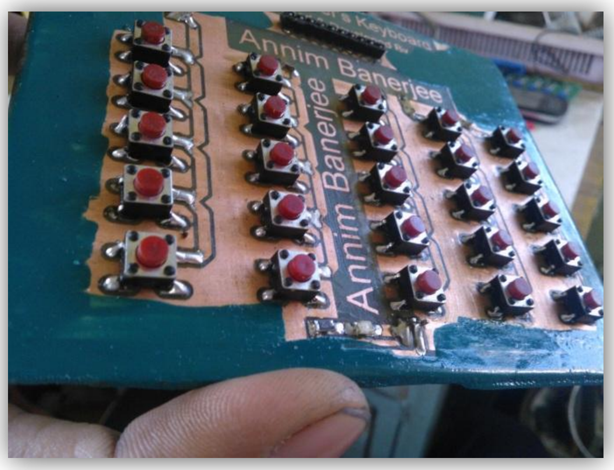

Start
Preparation
Connect
Connected?
NO
YES
Thread which Read data from COM port
NO
User Input
Interactive Display
Message pump
Data received from Programmer's Keyboard
YES
Jump
DisConnect
Kill Thread
Predefined Process To Exit
Apply keys SendInput/SenKeys API Call
Exit

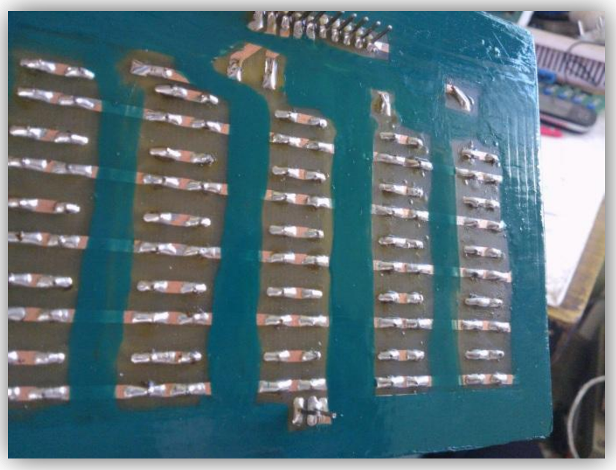
The above shown diagram is the control flow of the software which resides on PC and monitors the data coming from COM port from our *Programmer's Keyboard device*. There are some preparations process needed after starting the program. A message loop or say pump which is been provided by OS itself will monitor all user activities performed on screen which is called as interactive display unit. Connect or disconnect are the options by which they can get connect to device.

5. Results & Discussion

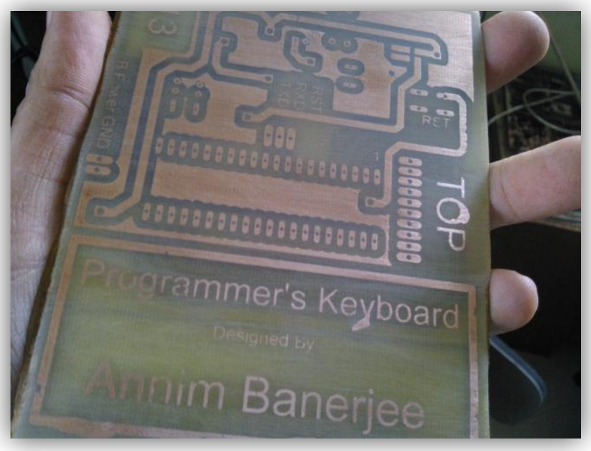
5.1 OUTPUT SCREENS

In this segment you will find some of the snaps of the hardware and working of the software which will be running on PC end. So, starting from hardware…



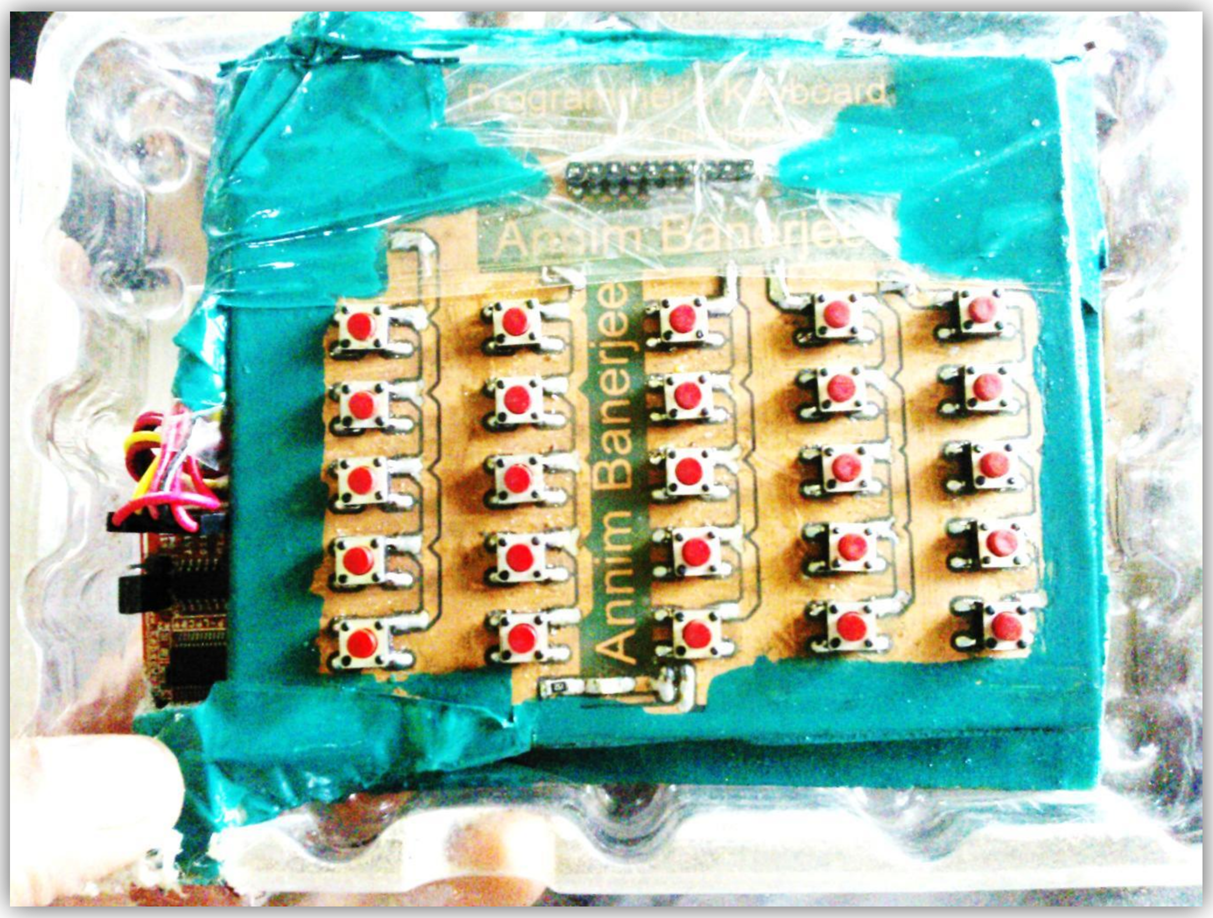


To the top, it is the TOP layer of keyboard part, of hardware and to the next down; it's the BOTTOM layer of the same. After fabrication, it has been painted by fabric paint of green in colour.

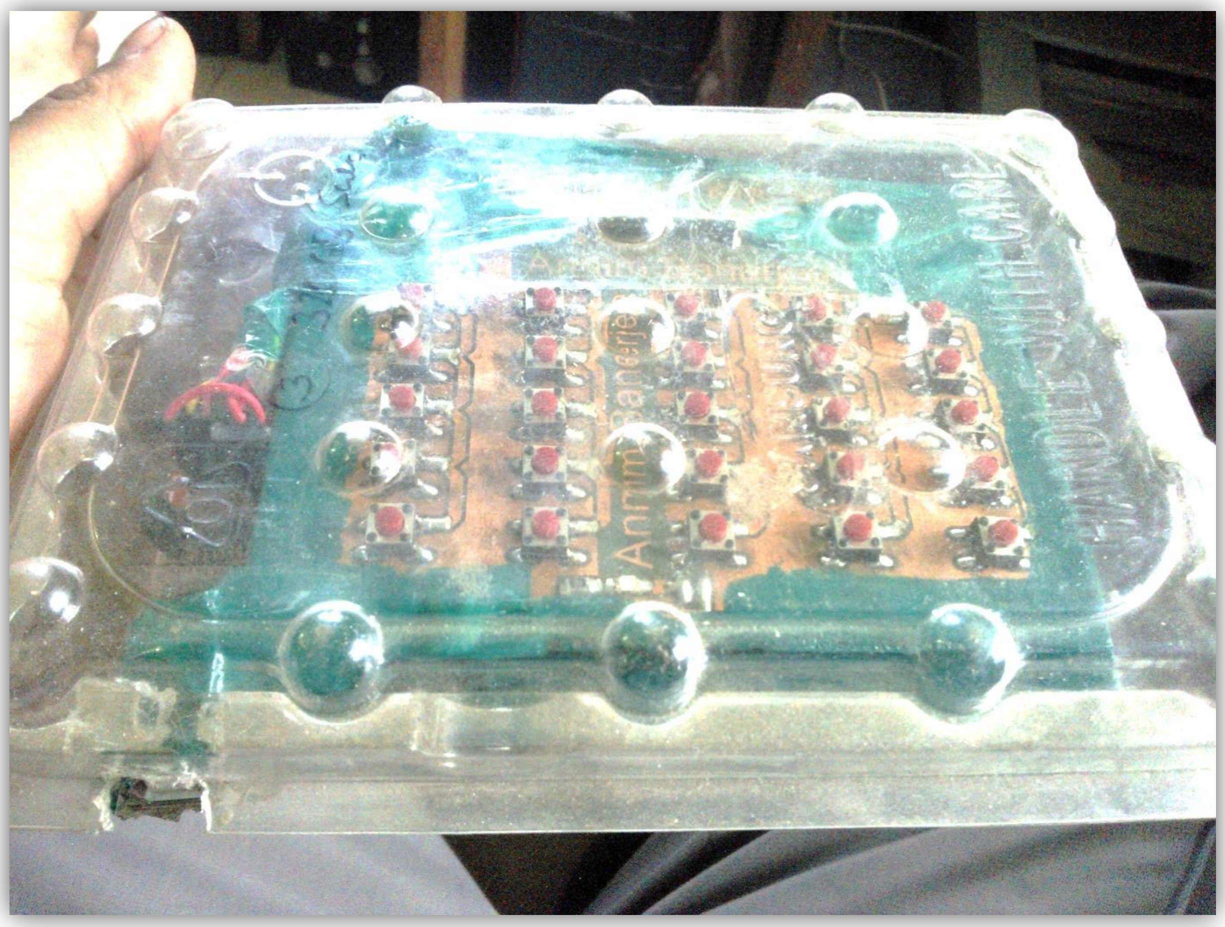


This snap shown to right down side, is the TOP layer of the controller board, of the hardware. This will be attached to the keyboard module, and keyboard module will sit on top of the controller board.

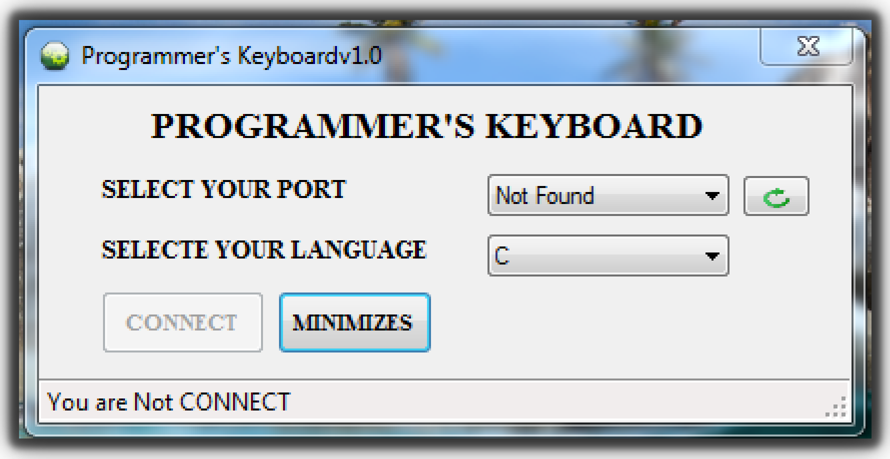
The combined, fully attached hardware snap shot is been shown below…



The above image is a top view of our keyboard hardware which is been packed in a HDD case and been painted again with fabric paint for preventing the hardware from rusting. Further for putting a rigidity or say firmness, we have taped it with packaging tape. Some more snaps of the final assembled hardware shown below…



The hardware is then tested and verified as per the requirement is been created. You can assume that BBT test is been done and our project is been succeeded in that test. Further on, software side, you have already seen some snaps of software which is been created for this hardware, but few more snaps are shown below which are developed in Dot.NET platform for the same …



Not Found

C

You are Not CONNECT

After when the device is connected and COM port is found, you can connect to device a on minimizing this window it will go to notification tray as notifying icon. And from there you can do you any changes by using the context menu options.

And when your work is done you can disconnect the device and can remove it safely.

6. Conclusion & Scope of work

6.1 CONCLUSION

Putting all together, in this segment-

- A keyboard with USB interfacing for connectivity with computer.
- 25 keys with separate functionalities to each, as much as possible.
- Default maximum rate of communication between device and computer is at 9600 baud rate.
- Matrix key logic is been implemented to make keyboard.
- Hardware front-
    - Divided into two module-
        - Keyboard module.
            - Separate board which got all the keys mounted on it. And a power led too.
        - Controller module.
            - Board on which the controller is been planted with USB interfacing module too attached with it.
    - Firmware code-
        - Program written in Embedded C.
        - To burn code- Flash Magic is used, provided by NXP.
    - Controller used-
        - NXP P89V51RD2BN/FN
- Software front-
    - On PC front
        - Software is been developed in two versions-
            - One written in C language using Win APIs.
            - Other is written in C#, implemented on Dot.NET platform v3.5.
- Benefits
    - This keyboard puts most of the MRU type code snippets on editor window then and there on one push of switch.
    - Act as plugin to those IDEs which didn’t got an Auto Completion feature.

❖ **Scope of work**

- The USB module should be included in the circuitry of controller board with best suitable USB-to-Serial converter chip.
- Can be converted into a normal workable keyboard with 102, 104 keys.
- Can be converted into a controller for specific game.
- A good logic should be developed to replace the controller, i.e. MCU from the project and also UART communication can be re-designed by using very simple TTL logics. A small mechanism is been shown which might replace the MCU and

also the USB-to-Serial converter chip, but this is an experimental, theoretical piece of information, shown in next segment…

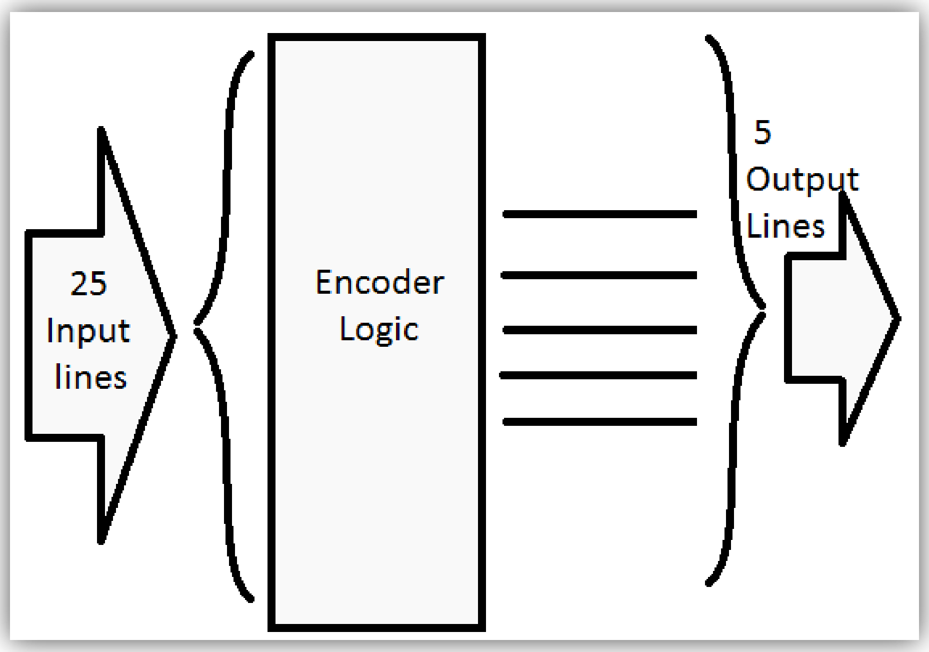
- **A theory**
  - To replace MCU from board

    To replace MCU from board we have to go with TTL logics as for now this is one of the ways to go and deal with. For this we have to recall the conditions & inputs and outputs.

    We have-

    - 25 keys-25 input lines.
    - Take 5 bits to express 25 in binary form.
    - Hence, 5 output lines.

    Thus, we have to deduce a kind of encoder logic which encodes 25 input lines to 455 output lines all together, iff one key is pressed at a time. We have to get a truth table for this to deduce an expression for our encoder logic.



In this logic diagram shown above, there are many instances where many combinations in output lines will be not used as in 5 bit, at max value could be 32, but here only 25 values or say combinations will be considered as valid ones.

Consider input value variable names: A-Y

Consider output value variable names: O1, O2, O3, O4, O5.

Now let's construct the truth table for the above encoder logic and then after we shall deduce an expression from truth table.

Input Variables																									**Output Variables**						**Dec**
LSB																								**MSB**		**MSB**				**LSB**	
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**		**O1**	**O2**	**O3**	**O4**	**O5**	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	1	0	2
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	1	1	3
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1	0	0	4
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1	0	1	5
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1	1	0	6
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1	1	1	7
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1	0	0	0	8
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1	0	0	1	9
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1	0	1	0	10
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1	0	1	1	11
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1	1	0	0	12
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		0	1	1	0	1	13
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0		0	1	1	1	0	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0		0	1	1	1	1	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		1	0	0	0	0	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		1	0	0	0	1	17
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0		1	0	0	1	0	18
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0		1	0	0	1	1	19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0		1	0	1	0	0	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	0	1	0	1	21
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		1	0	1	1	0	22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		1	0	1	1	1	23
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		1	1	0	0	0	24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		1	1	0	0	1	25

From the above truth table of 25 variables and 5 output variables, we shall deduce the Boolean expression one by one…

O1 = A+C+E+G+I+K+M+O+Q+S+U+W+Y.

O2= B+C+F+G+J+K+N+O+R+S+V+W.

O3= D+E+F+G+L+M+N+O+P+T+U+V+W.

O4= H+I+J+K+L+M+N+O+X+Y.

O5= P+Q+R+S+T+U+V+W+X+Y.

Where + means OR logic operation.

The truth table so deduced may or may not be the best optimal one. As this logic is getting more complex unlike it appears to. To calculate numbers of OR gates shall be required would be, 53 OR gates. Now if we pick a quad 2-OR gate IC, then around 14 OR gate chips are required. This may increase the cost of board and also consume space. This is because we are considering 25 keys here…

Now the data transmission between this logic to computer, this is a real challenge where a highly equipped machine has to communicate with our hardware. Available interfacing technologies are-

- USB v1.1, v2.0, v3.0, HS etc.
- Serial port as RS232, i.e. DB9 port with 9 pins.
- Serial Port as printer port, i.e. DB25 with 25 pins.

- **Protocol** to call for communication to happen-**using USB technology…**
    - We got one D+ pin and D- pin, one pin got data and other one is the mirror image of the same in negative mean.
    - Few more hardware required –
        - One 8-bit parallel in, serial out shift register chip-
            - This takes 5 input lines parallel and put the same data out in serial way.
            - Still 3 bits left, consider as *don't care*.
            - Requires a clock tick to push serially data out from chip towards computer port and get it register on computer's port buffer.
    - Case-
        - Computer peeks onto port buffer for any data except 0x00f.
        - Iff any button pressed-
            - For once and for first time, a high signal shall be issued over D+ line and negative of the same too over D-.
            - The point when the software program finds a change in port buffer, it then recognised as ***START*** of the communication.
            - XOR the computer's port buffer with 0x00f data.
            - For next 8 cycles…
                - From computer, D+ carries a dummy HIGH signal to give a clock to parallel-in-serial-out sift register and over same channel D+, one by one bit will be transferred in to computers port buffer by ORing the value with previously stored data in computer's port buffer.

        - The final value so obtained in the computer's port buffer will be in binary format indicating the nth number of key is been pressed!
        - Immediately XOR the computer's port buffer with 0x00f for clearing the buffer.
    - Put a voltage inverter will put the mirror of D+ in D- BUS line for correct communication as per USB data communication protocol.
    - Place a fuse on D+ and D- line for eliminating noise and have strong strength of data in BUS lines.

- **Protocol** to call for communication to happen-**using Serial port DB9, RS232 technology…**
    - With this port, we got one more signal BUS line and that is DTR signal which stands for ***Detect-To-Ready*** signal which acts as ***RESET*** signal for any MCU or any digital system. NOTing to this signal will be better in terms of usage.
    - Case regarding this technology is quite similar to USB technology based theory-protocol. We shall send a DUMMY ***START*** signal to computer from DTR BUS line and over software end, program will constantly pulling LOW. If found HIGH, that indicates a START condition and rest goes the same. This DTR will be used to give a clock tick to parallel-in-serial-out shift register to get the data bits out from it.
    - The TXD line of DB9 pin shall be used to put data from keyboard to computer's port buffer.
    - A voltage doubler of +10V circuit will be required to put data which will be evaluated for validation purpose by the PC.

A practical implementation of this theory may leads to an opportunity for creating a good and new technique for communication between a simple digital IO boards to computer. Although, the theory and protocol so discussed above, require a recitation from an experts of this domain. This is purely experimental, information content. This information can be shared, changed, distributable on your own risk. There are lot more work left which turns as scope of work to do.

Bibliography

Books:

- The 8051 Microcontroller And Embedded Systems Using Assembly And C, 2/E
    - ISBN: 8131710262, 9788131710265

Links:

- http://pcbheaven.com/wikipages/How_Key_Matrices_Works/
- http://www.learningaboutelectronics.com/Articles/How-does-a-matrix-keyboard-scanning-algorithm-work
- http://www.engineersgarage.com/tutorials/how-to-make-a-pcb-at-home
- http://www.ingeniumblog.net/2010/06/20-best-pcb-making-techniques/
- http://www.embeddedmarket.com
- http://www.onlinetps.com
- http://vishaworld.com
- http://www.flashmagictool.com/
- http://www.esacademy.com/blog/category/flashmagic/
- http://www.google.co.in/url?sa=t&rct=j&q=ft232rl&source=web&cd=1&cad=rja&sqi=2&ved=0CC4QFjAA&url=http%3A%2F%2Fwww.ftdichip.com%2FProducts%2FICs%2FFT232R.htm&ei=agZCUd7XC5CsrAfmj4GIDg&usg=AFQjCNGpVGg1DjvkQO3rhqGOpB-n6jA83A&bvm=bv.43287494,d.bmk
- http://www.google.co.in/url?sa=t&rct=j&q=ft232rl&source=web&cd=2&cad=rja&sqi=2&ved=0CDMQFjAB&url=http%3A%2F%2Fwww.ftdichip.com%2FDocuments%2FDataSheets%2FICs%2FDS_FT232R.pdf&ei=agZCUd7XC5CsrAfmj4GIDg&usg=AFQjCNGami1T1YXLF1U9kCK2hDTudVhVnQ&bvm=bv.43287494,d.bmk

Above are some links which are our source of information and based on which we have developed the ***Programmer's Keyboard*** project.