1. INTRODUCTION

OBJECTIVE : Main objective of this project to design a Graphical User Interface using VC++ that enables the user to configure the available ports/pins for input/output. The GUI designed controls the status of the pins present on the board and straightaway generates a dynamic C code that can be used directly to modify the pin status as per the user's requirement.

1.1. The BL2600

Selection of BL2600 as a reasonable option for implementation of our project

- Fast time to use it as a fully engineered, "ready-to-run/read-to-program" microprocessor core.
- Competitive pricing when compared with the alternative of purchasing and assembling individual components.
- Easy C-language program development and debugging(use of dynamic C)
- Generous memory size allows large programs with tens of thousands of lines of code, and substantial data storage.
- Reference design allows integrated Ethernet port for network connectivity, with easy to implement TCP/IP software.

The BL2600 is a C-programmable singleboard computer that has built-in digital and analog I/O combined with Ethernet connectivity. A Rabbit 3000 microprocessor operating at 44.2 MHz provides fast data processing with 10/100Mbps Ethernet connectivity. The BL2600 incorporates Rabbit 3000 microprocessor, flash memory, serial flash, static RAM, digital I/O ports, A/D converter inputs, D/A converter outputs, RS-232/RS-485 serial ports, and a 10/100Mbps Ethernet port.

Features of BL2600:

- Rabbit 3000 microprocessor operating at 44.2 MHz.
- 512K static RAM and 512K flash memory standard.
- 36 digital I/O: 16 protected digital inputs, 4 high-current digital outputs software configurable as sinking or sourcing, and 16 I/O individually software-configurable as inputs or sinking outputs.
- 12 analog channels: eight 11-bit A/D converter inputs, four 12-bit D/A converter 0–10 V or ±10 V buffered outputs.
- One RJ-45 Ethernet port compliant with IEEE 802.3 standard for 10/100Mbps Ethernet protocol.
- Three Ethernet status LEDs .
- Three serial ports (2 RS-232 or 1 RS-232 with RTS/CTS, 1 RS-485 or RS-232).
- Two Rabbit Net expansion ports
- Battery-backed real-time clock.

2. The Rabbit 3000 Microprocessor

The BL2600 runs on the Rabbit 3000 microprocessor. The Rabbit 3000 has been designed by a long-time manufacturer of low-cost single-board computers and are supported by an innovative C-language development system (Dynamic C). The Rabbit has outstanding computation speed for a microprocessor with an 8-bit bus. Microprocessor hardware and software development is easy for Rabbit users. Software development is accomplished by connecting a simple interface cable from a PC serial port to the Rabbit-based target system or by performing software development and debugging over a network or the Internet using interfaces and tools provided by Rabbit Semiconductor.

2.1 Features and Specifications Rabbit 3000

- 128-pin LQFP package. Operating voltage 1.8 V to 3.6 V. Clock speed to 54 MHz. All specifications are given for both industrial and commercial temperature and voltage ranges. Rabbit microprocessors are low-cost.
- Industrial specifications are for 3.3 V ±10% and a temperature range from -40°C to +85°C. Modified commercial specifications are for a voltage variation of 5% and a temperature range from -40°C to 70°C.
- 1-megabyte code-data space allows C programs with 50,000+ lines of code. The extended Z80-style instruction set is C-friendly, with short and fast opcodes for the most important C operations.
- Four levels of interrupt priority make a fast interrupt response practical for critical applications. The maximum time to the first instruction of an interrupt routine is about 0.5 µs at a clock speed of 50 MHz.
- Access to I/O devices is accomplished by using memory access instructions with an I/O prefix. Access to I/O devices is thus faster and easier compared to processors with a distinct and narrow I/O instruction set. As an option the auxiliary I/O bus can be enabled to use separate pins for address and data, allowing the I/O bus to have a greater physical extent with less EMI and less conflict with the requirements of the fast memory bus.(Further described below.)
- Hardware design is simple. Up to six static memory chips (such as RAM and flash memory) connect directly to the microprocessor with no glue logic. A memory-access time of 55 ns suffices to support up to a 30 MHz clock with no wait states; with a 30 ns memory-access time, a clock speed of up to 50 MHz is possible with no wait states. Most I/O devices may be connected without glue logic.
- The memory read cycle is two clocks long. The write cycle is 3 clocks long. A clean memory and I/O cycle completely avoid the possibility of bus fights. Peripheral I/O devices can usually be interfaced in a glueless fashion using the common /IORD and /IOWR strobes in addition to the user-configurable IO strobes on Parallel Port E. The Parallel Port E pins can be configured as I/O read, write, read/write, or chip select when they are used as I/O strobes.
- EMI reduction features reduce EMI levels by as much as 25 dB compared to other similar microprocessors. Separate power pins for the on-chip I/O buffers prevent

high-frequency noise generated in the processor core from propagating to the signal output pins. A built-in clock spectrum spreader reduces electromagnetic interference and facilitates passing EMI tests to prove compliance with government regulatory requirements. As a consequence, the designer of a Rabbit-3000-based system can be assured of passing FCC or CE EMI tests as long as minimal design precautions are followed.

- The Rabbit may be cold-booted via a serial port or the parallel access slave port. This means that flash program memory may be soldered in unprogrammed, and can be reprogrammed at any time without any assumption of an existing program or BIOS. A Rabbit that is slaved to a master processor can operate entirely with volatile RAM, depending on the master for a cold program boot.
- There are 56 parallel I/O lines (shared with serial ports). Some I/O lines are timer synchronized, which permits precisely timed edges and pulses to be generated under combined hardware and software control. Pulse-width modulation outputs are implemented in addition to the timer-synchronization feature (see below).
- Four pulse width modulated (PWM) outputs are implemented by special hardware. The repetition frequency and the duty cycle can be varied over a wide range. The resolution of the duty cycle is 1 part in 1024.
- There are six serial ports. All six serial ports can operate asynchronously in a variety of commonly used operating modes. Four of the six ports (designated A, B, C, D) support clocked serial communications suitable for interfacing with "SPI" devices and various similar devices such as A/D converters and memories that use a clocked serial protocol. Two of the ports, E and F, support HDLC/SDLC synchronous communication. These ports have a 4-byte FIFO and can operate at a high data rate. Ports E and F also have a digital phase-locked loop for clock recovery, and support popular data-encoding methods. High data rates are supported by all six serial ports. The asynchronous ports also support the 9th bit network scheme as well as infrared transmission using the IRDA protocol. The IRDA protocol is also supported in SDLC format by the two ports that support SDLC.
- A slave port allows the Rabbit to be used as an intelligent peripheral device slaved to a master processor. The 8-bit slave port has six 8-bit registers, 3 for each direction of communication. Independent strobes and interrupts are used to control the slave port in both directions. Only a Rabbit and a RAM chip are needed to construct a complete slave system, if the clock and reset control are shared with the master processor
- There is an option to enable an auxiliary I/O bus that is separate from the memory bus. The auxiliary I/O bus toggles only on I/O instructions. It reduces EMI and speeds the operation of the memory bus, which only has to connect to memory chips when the auxiliary I/O bus is used to connect I/O devices. This important feature makes memory design easy and allows a more relaxed approach to interfacing I/O devices.
- The built-in battery-backable time/date clock uses an external 32.768 kHz crystal oscillator. The suggested model circuit for the external oscillator utilizes a single "tiny logic" active component. The time/date clock can be used to provide

periodic interrupts every 488 µs. Typical battery current consumption is about 3 µA.

- Numerous timers and counters can be used to generate interrupts, baud rate clocks, and timing for pulse generation.
- Two input-capture channels can be used to measure the width of pulses or to record the times at which a series of events take place. Each capture channel has a 16-bit counter and can take input from one or two pins selected from any of 16 pins.
- Two quadrature decoder units accept input from incremental optical shaft encoders. These units can be used to track the motion of a rotating shaft or similar device.
- A built-in clock doubler allows ½-frequency crystals to be used.
- The built-in main clock oscillator uses an external crystal or a ceramic resonator. Typical crystal or resonator frequencies are in the range of 1.8 MHz to 30 MHz. Since precision timing is available from the separate 32.768 kHz oscillator, a low-cost ceramic resonator with ½ percent error is generally satisfactory. The clock can be doubled or divided down to modify speed and power dynamically. The I/O clock, which clocks the serial ports, is divided separately so as not to affect baud rates and timers when the processor clock is divided or multiplied. For ultra low power operation, the processor clock can be driven from the separate 32.768 kHz oscillator and the main oscillator can be powered down. This allows the processor to operate at approximately between 20 and 100 µA and still execute instructions at the rate of up to 10,000 instructions per second. The 32.768 kHz clock can also be divided by 2, 4, 8 or 16 to reduce power. This "sleepy mode" is a powerful alternative to sleep modes of operation used by other processors.
- Processor current requirement is approximately 65 mA at 30 MHz and 3.3 V. The current is proportional to voltage and clock speed--at 1.8 V and 3.84 MHz the current would be about 5 mA, and at 1 MHz the current is reduced to about 1 mA.
- To allow extreme low power operation there are options to reduce the duty cycle of memories when running at low clock speeds by only enabling the chip select for a brief period, long enough to complete a read. This greatly reduces the power used by flash memory when operating at low clock speeds.
- The excellent floating-point performance is due to a tightly coded library and powerful processing capability. For example, a 50 MHz clock takes 7 µs for a floating add, 7 µs for a multiply, and 20 µs for a square root. In comparison, a 386EX processor running with an 8-bit bus at 25 MHz and using Borland C is about 20 times slower.
- There is a built-in watchdog timer.
- The standard 10-pin programming port eliminates the need for in-circuit emulators. A very simple 10-pin connector can be used to download and debug software using Z-World's Dynamic C and a simple connection to a PC serial port. The incremental cost of the programming port is extremely small.

The Rabbit 3000 is an evolutionary design. The Rabbit has no special support for dynamic RAM but it has extensive support for static memory. This is because the price of static memory has decreased to the point that it has become the preferred choice for medium-scale embedded systems.

The Rabbit has no support for DMA (direct memory access) because most of the uses for which DMA is traditionally used do not apply to embedded systems, or they can be accomplished better in other ways, such as fast interrupt routines, external state machines or slave processors. The main problem is the lack of instructions for handling 16-bit words and for accessing data at a computed address, especially when the stack contains that data.

There are six serial ports designated ports A, B, C, D, E, and F. All six serial ports can operate in an asynchronous mode up to a baud rate equal to the system clock divided by 8. The asynchronous ports use 7-bit or 8-bit data formats, with or without parity. A 9th bit address scheme, where an additional bit is set or cleared to mark the first byte of a message, is also supported. There are 56 parallel input/output lines divided among seven 8-bit ports designated A through G. Most of the port lines have alternate functions, such as serial data or chip select strobes. Parallel Ports D, E, F, and G have the capability of timer-synchronized outputs.

The BIOS is provided with Dynamic C and works for all Rabbit controlled processors. The BIOS is compiled separately from the user's application. It occupies space at the bottom of the root code segment. When execution of the user's program starts at address zero on power-up or reset, it starts in the BIOS. When Dynamic C cold-boots the target and downloads the binary image of the BIOS, the BIOS symbol table is retained to make its entry points and global data available to the user application. Board specific drivers are compiled with the user's program after the BIOS is compiled.

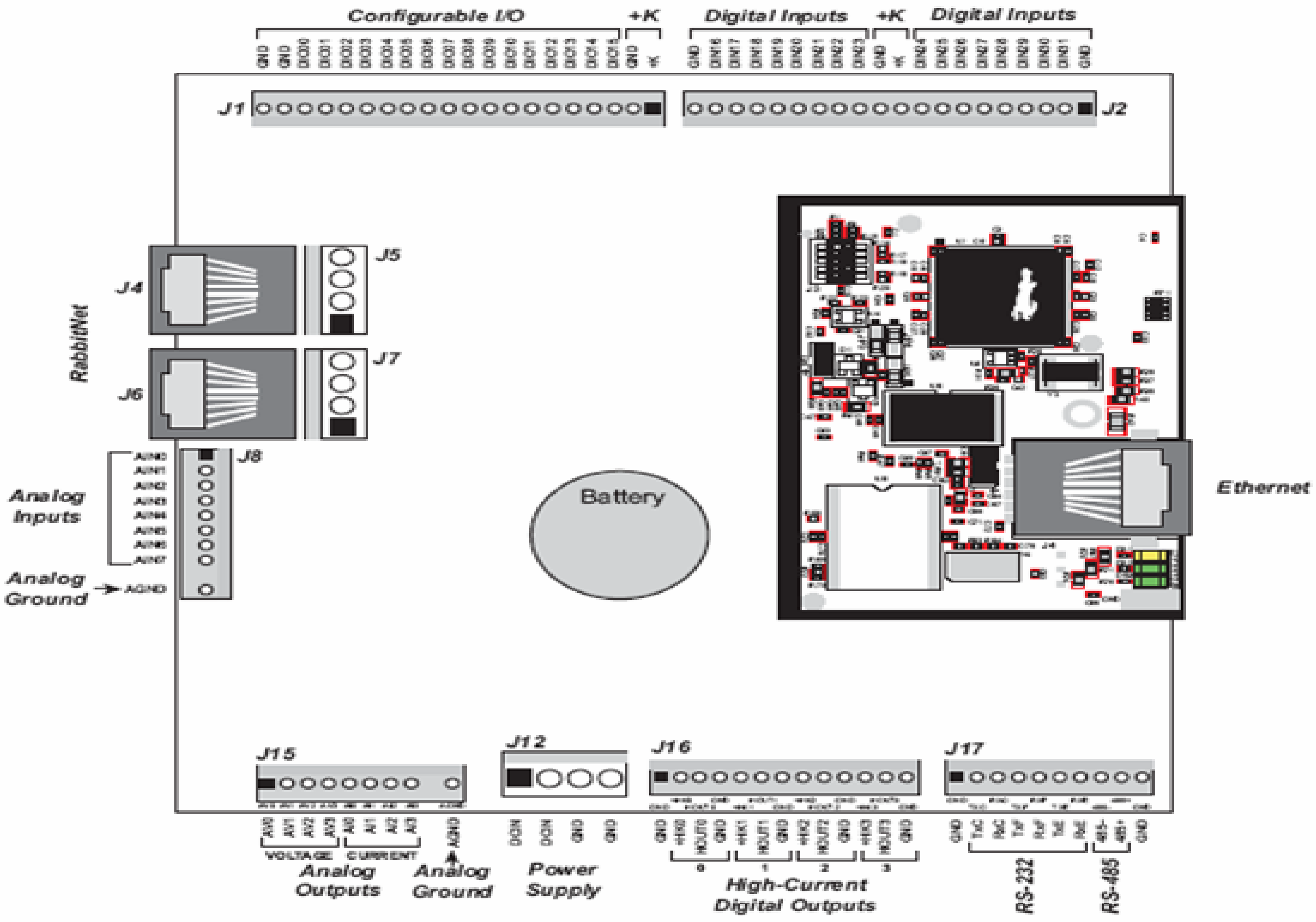
The Rabbit may be cold-booted via a serial port or the parallel access slave port. This means that flash program memory may be soldered in unprogrammed, and can be reprogrammed at any time without any assumption of an existing program or BIOS.

3. Programming the BL2600:

Dynamic C is an IDE for writing embedded software. It is designed for use with controllers based on the Rabbit microprocessor. The Rabbit families of processors are high-performance 8-bit microprocessors that can handle C language applications of approximately 50,000 C+ statements or 1 MB.
Dynamic C integrates development functions like Editing, Compiling, Linking, Loading, Debugging into one program. In fact, compiling, linking and loading are one function. Dynamic C programs can be executed and debugged interactively at the source-code or machine-code level. Dynamic C also supports assembly language programming.
Dynamic C provides extensions to the C language (such as shared and protected variables, co-statements and co-functions) that support real-world embedded system development. Dynamic C supports cooperative and preemptive multitasking. Dynamic C comes with many function libraries, all in source code. These libraries support realtime programming, machine level I/O, and provide standard string and math functions.
Dynamic C has a set of features that allow the programmer to make fullest use of extended memory. Dynamic C supports the 1 MB address space of the microprocessor. The address space is segmented by a memory management unit (MMU). There are several differences between C and dynamic C such as if a variable is explicitly initialized in a declaration (e.g., int x = 0;), it is stored in flash memory (EEPROM) and cannot be changed by an assignment statement. Such a declaration will generate a warning that may be suppressed using the const keyword or When declaring pointers to functions, arguments should not be used in the declaration and also bit fields are not supported.

Many functional features such as function chaining (allows special segments of code to be embedded within one or more functions. When a named function chain executes, all the segments belonging to that chain execute), Co-statements (allow concurrent parallel processes to be simulated in a single program), Co-functions (allow cooperative processes to be simulated in a single program) & Slice statements (allow preemptive processes in a single program) are also available.

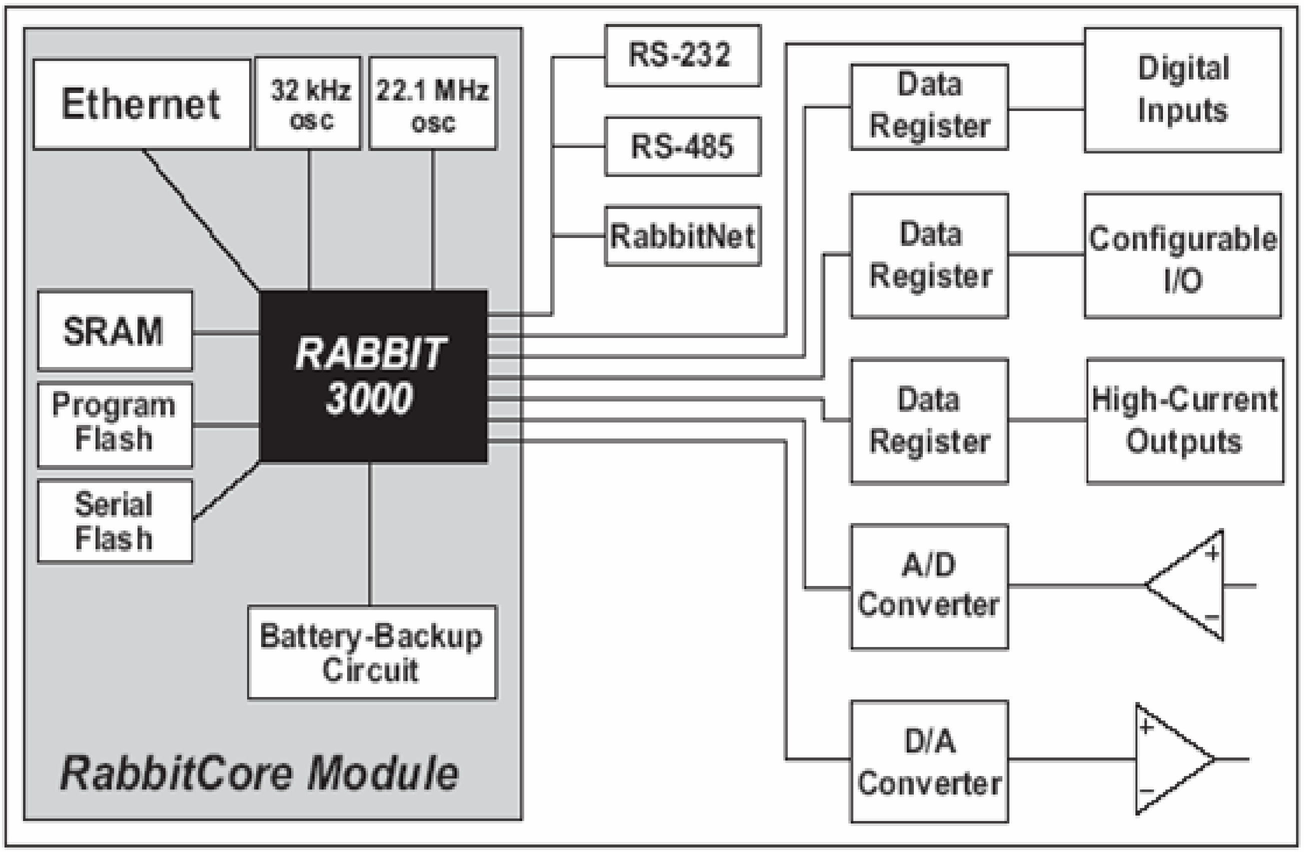
BL 2600 Pin Out Diagram



4. Subsystems integrated in BL 2600

Subsystems integrated in BL 2600 are :

- Digital I/O.
- Serial Communication.
- A/D Converter Inputs.
- D/A Converter Outputs.
- Analog Reference Voltage Circuit.



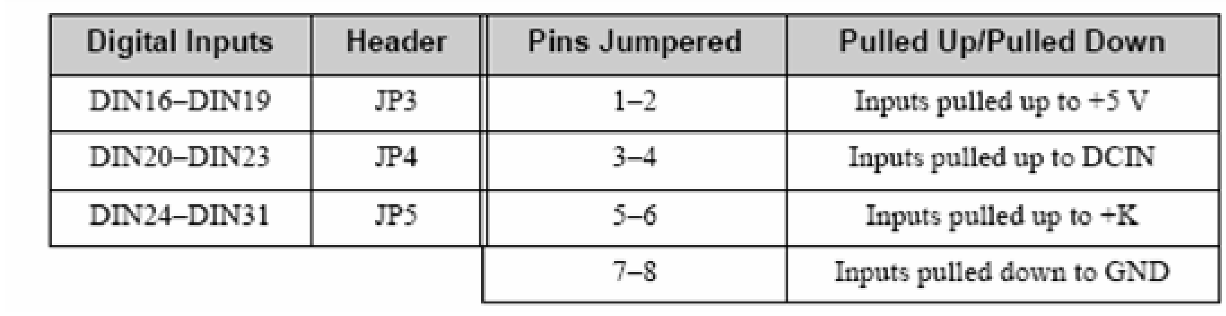
4.1 Digital Input/Output:

- Digital Inputs.
- Configurable I/O.
- High Current Digital Outputs.

(a) Digital Inputs

The BL2600 has 16 digital inputs, DIN16–DIN31DIN31, each of which is protected over a range of –36 V to +36 V. The inputs b default are configured to be pulled up to +5 V, but they can also be pulled up to +K or DCIN, or pulled down to 0

V in banks by changing a jumper as shown in the table below i.e. the various jumper positions and the corresponding reference voltages .
Dynamic C allows the use of digIn function to call to read the digital inputs, DIN16–DIN31 and are considered to be digital input channels 16–31.The actual switching threshold is approximately 1.40 V. Anything below this value is a logic 0, and anything above is a logic 1. The digital inputs are each fully protected over a range of -36 V to +36 V, and can handle short spikes of ±40 V. Individual DIN16–DIN23 channels may be used for interrupts, input capture, as quadrature decoders, or as PWM outputs.



Digital Inputs	Header	Pins Jumpered	Pulled Up/Pulled Down
DIN16–DIN19	JP3	1–2	Inputs pulled up to +5 V
DIN20–DIN23	JP4	3–4	Inputs pulled up to DCIN
DIN24–DIN31	JP5	5–6	Inputs pulled up to +K
		7–8	Inputs pulled down to GND

(b) High-Current Digital Outputs & PWM Outputs
The BL2600 has four high-current digital outputs, HOUT0–HOUT3, which can each sink or source up to 2A. When the BL2600 is first powered up or reset, all the outputs are disabled, that is, at a high-impedance state, until the digHoutConfig software function call is made. The digHoutConfig call sets the initial state of each high-current output according to the configuration specified by the user, and enables the digital outputs to their initial status. Digital inputs DIN20–DIN23 can be used as PWM output channels by setting the jumper on header JP4 across pins 7–8 to pull the digital inputs to ground.

(c) Configurable I/O
The BL2600 has 16 configurable I/O that may be configured individually in software as either digital inputs or as digital outputs. By default, a configurable I/O channel is a digital input, but may be set as a digital output by using the digOutConfig function call.The software digIn function call is used to read the configurable I/O, DIO 00-DIO15 are considered to be digital input channels 00-15 (also digIn function call can also read these channels if they are set to be digital outputs).

4.2 Serial Communication

For Serial communication a total of four configurations are available using serMode software function call in Dynamic C header .The BL2600 has three serial communication ports (serial communication is made possible by the use of RS232 or RS485 channels existing on the board. The BL2600 also has one CMOS serial channel that serves as the programming port. All four serial ports operate in an asynchronous mode.

The CMOS channel or the programming port also supports the clocked or synchronous mode in which the clock is provided by BL 2600.

(a) RS-232 serial port

RS-232 serial communication is supported by an RS-232 transceiver and uses RS-232 serial communication protocol. The polarity is reversed in an RS-232 circuit so that a +3.3 V output becomes approximately -10 V and 0 V is output as +10 V. RS-232 can be used effectively at the BL2600's maximum baud rate for distances of up to 15 m.

(b) Programming Port

The Rabbit Core module on the BL2600 has a 10-pin programming header. The programming port uses the Rabbit 3000's Serial Port A for communication.The functions of programming port are programming/debugging & Cloning.

4.3 A/D Converter Inputs

The A/D converter chip has a programmable amplifier and each external input has circuitry that provides scaling and filtering. The single A/D converter chip used in the BL2600 has a resolution of 12 bits (11 bits for the value and one bit for the polarity). By pairing the analog inputs and setting the reference voltage from the D/A converter measurements are possible, and can be configured for each channel or channel pair with the opmode parameter in the anaInConfig software function call. Usually channels AIN0-AIN3 are used for current measurements. (this is achieved by setting corresponding jumper on header JP6).To get the best results form the A/D converter, it is necessary to calibrate each mode (single-ended, differential, and current) for each of its gains.

4.4 D/A Converter Output

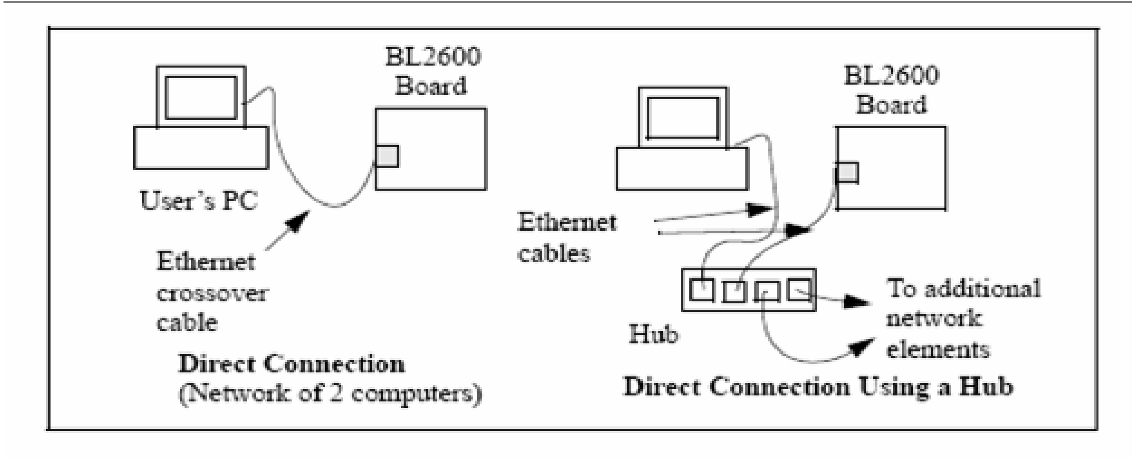
The four D/A converter outputs are buffered and scaled to provide an output from 0 V to +10 V (12-bit resolution) or ±10 V (11-bit resolution, one bit used for polarity). The voltage and the current outputs for a particular channel are driven by the same output on the D/A converter chip. As a result, either the anaOutVolts or the anaOutmAmps function calls will set both the voltage and the current outputs corresponding to a particular channel. Because of the "connection" between the analog voltage outputs and the analog current outputs, the configuration of the analog voltage outputs with the anaOutConfig function call as unipolar outputs with 12-bit resolution or as bipolar outputs with 11-bit resolution also affects the resolution current outputs as you need to configure a voltage output for unipolar operation if you want 12-bit resolution on the associated current output. The D/A converter outputs are factory-calibrated and the calibration constants are stored in a separate EEPROM.

4.5. Analog Reference Voltage Circuit

The A/D converter chip supplies the 2.048 V reference voltage, which is divided in half and then amplified and buffered to provide the 1.667 V and 2.5 V reference voltages used by the digital output circuits. The D/A converter chip provides the reference voltages for the digital inputs to provide single-ended unipolar or differential measurements. Because the D/A converter chip operation is configured by the anaOutConfig function, it is important to run the anaOutConfig function before running anaInConfig.

5. TCP/IP Features

Ethernet Connections



TCP is a connection-oriented transport service; it provides end-to-end reliability, re-sequencing ,and flow control.
TCP/IP stands for "Transfer Control Protocol/ Internet Protocol" was developed for end to end bit stream communication over an unreliable network (internet). TCP/IP independent of physical medium or Ethernet frames.

5.1 TCP/IP over Ethernet

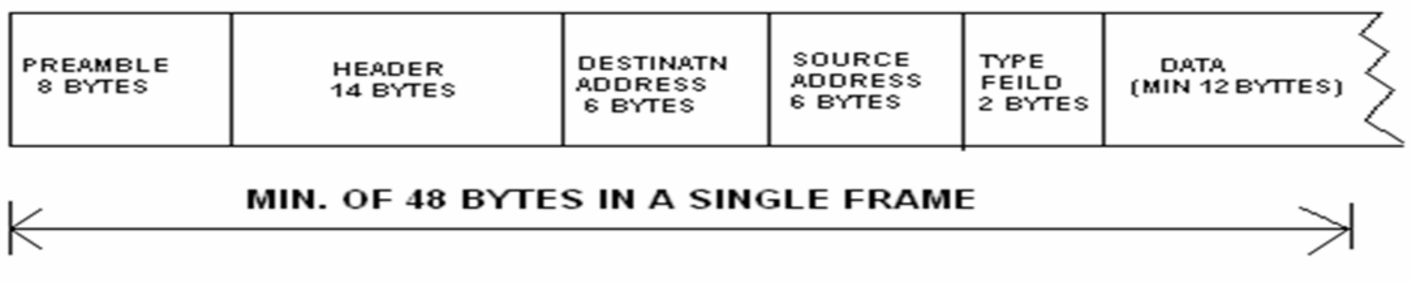
TCP/IP (Transmission Control Protocol/Internet Protocol) is a set of protocols independent of the physical medium used to transmit data, but most data transmission for Internet communication begins and ends with Ethernet frames.
The Ethernet can use either a bus or star topology. A bus topology attaches all devices in sequence on a single cable. In a star topology all devices are wired directly to a central hub. 10Base-T uses a combination called a star-shaped bus topology because while the attached devices can share all data coming in on the cable, the actual wiring is in a star shape.
The access method used by the Ethernet is called Carrier Sense Multiple Access with Collision Detect (CSMA/CD). This is a contention protocol, meaning it is a set of rules to follow when there is competition for shared resources.

Packets referred to as frames are sent over the network

A frame is designed as follows:

- PREAMBLE
- HEADER
- DESTINATION AND SOURCE ADDRESS
- TYPE FIELD
- DATA

5.2 TCP Features in Dynamic C



The main Library file is DCRTCP.lib. It consists of:

- TCP.lib
- IP.lib
- DNS.lib
- NET.lib

To use TCP features on BL2600:

- Make changes in the “tcp_config.lib” file.
- Define the TCPCONFIG macro with individual MY_IP_ADDRESS, MY_NETMASK, MY_GATEWAY, and MY_NAMESERVER macros in each program.
- Leaving TCPCONFIG at the default of 1, will set the IP configurations to 10.10.6.100, the netmask to 255.255.255.0, and the nameserver and gateway to 10.10.6.1.

5.3 The tcp_config file

/* the following file has been changed from the originally available file for generic use */

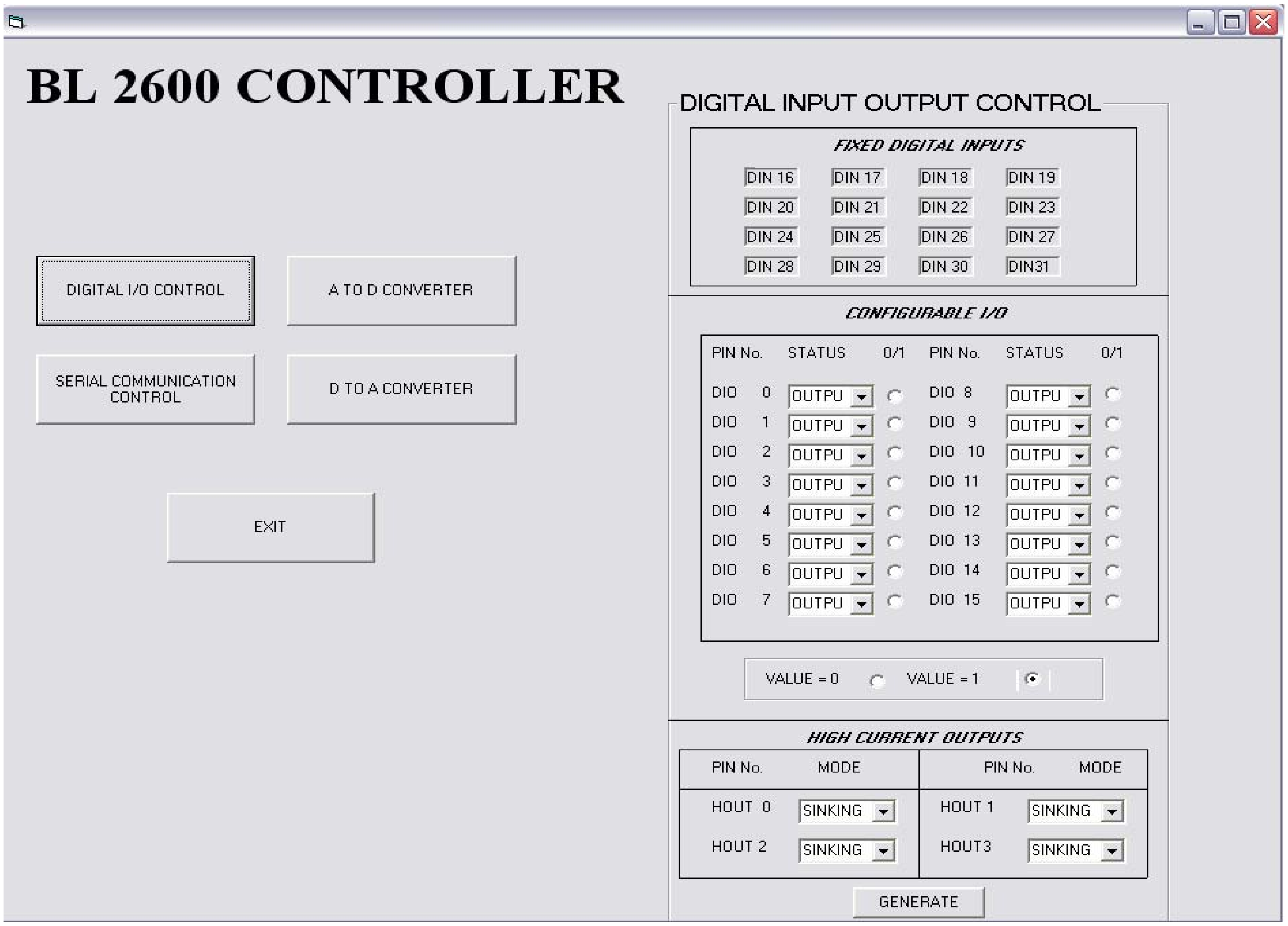
```
#ifndef TCPCONFIG
                #ifndef TCP_CONFIG
        #warnt "TCPCONFIG was not defined.  Defaulting to 0 (no predefined config)."
    #warnt "To get rid of this warning, explicitly #define TCPCONFIG 0."
                #define TCPCONFIG          0                // Zero means do not do any
configuration: expects
                                                                               //     all
config items to be defined in the main
                                                                               //
program.
        #else
                #define TCPCONFIG TCP_CONFIG
                #warnt "TCPCONFIG is the correct TCP/IP config macro. Using
TCP_CONFIG anyway..."
```

```
	#endif
#endif

#if TCPCONFIG >= 100
	#use "custom_config.lib"
#else

		#define _PRIMARY_STATIC_IP		"10.10.6.100"
	#define _PRIMARY_NETMASK		"255.255.255.0"
	#ifndef MY_NAMESERVER
		#define MY_NAMESERVER		"10.10.6.1"
	#endif
	#ifndef MY_GATEWAY
		#define MY_GATEWAY		"10.10.6.1"
	#endif

	#if TCPCONFIG == 1
		/*
		 * Config 1: Simple static configuration of single ethernet interface
		 */
		#define USE_ETHERNET		1
		#define IFCONFIG_ETH0 \
				IFS_IPADDR,aton(_PRIMARY_STATIC_IP), \
				IFS_NETMASK,aton(_PRIMARY_NETMASK), \
				IFS_UP
	#endif
	#endif
```

6. The GUI

The GUI was made using VC++. It incorporates the subsystems provided with the BL2600.

- There are 16 configurable Digital Input/Output pins referred to as DIO 0-DIO15.
- There are 16 fixed Digital Inputs which are referred to as DIN 16-DIN31.
- There are 4 High Current outputs that can be configured for either sourcing or sinking and are depicted as HOUT 0- HOUT3.
- There is a separate column for selecting the channel pair and the opmode working condition for the A to D converter.
- There is another column for the selection of the configuration and the mode of operation of the D to A converter.



PIN No.	STATUS	0/1	PIN No.	STATUS	0/1
DIO 0	OUTPU		DIO 8	OUTPU	
DIO 1	OUTPU		DIO 9	OUTPU	
DIO 2	OUTPU		DIO 10	OUTPU	
DIO 3	OUTPU		DIO 11	OUTPU	
DIO 4	OUTPU		DIO 12	OUTPU	
DIO 5	OUTPU		DIO 13	OUTPU	
DIO 6	OUTPU		DIO 14	OUTPU	
DIO 7	OUTPU		DIO 15	OUTPU	

VALUE = 0 [✓ VALUE = 1]

PIN No.	MODE	PIN No.	MODE
HOUT 0	SINKING	HOUT 1	SINKING
HOUT 2	SINKING	HOUT3	SINKING

7. Dynamic C

Dynamic C is an integrated development system for writing embedded software. It is designed for use with Z-World controllers and other controllers based on the Rabbit microprocessor. The Rabbit family of processors are high-performance 8-bit microprocessors that can handle C language applications of approximately 50,000 C+ statements or 1 MB.

7.1 Dynamic C features

Dynamic C integrates the following development functions:

- Editing
- Compiling
- Linking
- Loading
- Debugging

Dynamic C has an easy-to-use, built-in, full-featured, text editor. Debugging under Dynamic C includes the ability to use printf commands, watch expressions and breakpoints. Watch expressions can be used to compute C expressions involving the target's program variables or functions. Watch expressions can be evaluated while stopped at a breakpoint or while the target is running its program. Dynamic C provides extensions to the C language (such as shared and protected variables, co-statements and co-functions) that support real-world embedded system development. Dynamic C supports cooperative and preemptive multitasking. Dynamic C comes with many function libraries, all in source code. These libraries support real-time programming, machine level I/O, and provide standard string and math functions. Dynamic C compiles directly to memory. Functions and libraries are compiled and linked and downloaded on-the-fly. On a fast PC, Dynamic C might load 30,000 bytes of code in 5 seconds at a baud rate of 115,200 bps.

7.2 Dynamic C Enhancements

Many enhancements of Dynamic C are:

Function chaining, a concept unique to Dynamic C, allows special segments of code to be embedded within one or more functions. When a named function chain executes, all the segments belonging to that chain execute. Function chains allow software to perform initialization, data recovery, or other kinds of tasks on request.

- Costatements allow concurrent parallel processes to be simulated in a single program.
- Cofunctions allow cooperative processes to be simulated in a single program.
- Slice statements allow preemptive processes in a single program.
- Dynamic C supports embedded assembly code and stand-alone assembly code.
- Dynamic C has shared and protected keywords that help protect data shared between different contexts or stored in battery-backed memory.
- Dynamic C has a set of features that allow the programmer to make fullest use of extended memory. Dynamic C supports the 1 MB address space of the microprocessor. The address space is segmented by a memory management unit (MMU). Normally, Dynamic C takes care of memory management, but there are instances where the programmer will want to take control of it. Dynamic C has keywords and directives to help put code and data in the proper place.

7.3 Dynamic C Differences

The main differences in Dynamic C are

- If a variable is explicitly initialized in a declaration (e.g., `int x = 0;`), it is stored in flash memory (EEPROM) and cannot be changed by an assignment statement. Such a declaration will generate a warning that may be suppressed using the const keyword:
- const int x = 0
- To initialize static variables in Static RAM (SRAM) we use #GLOBAL_INIT sections entering the main function. Because Dynamic C has a library system that automatically provides function prototypes and

similar header information to the compiler before the user's program is compiled. This is done via the `#use` directive. This is an important topic for users who are writing their own libraries. Separate compilation of different parts of the program is not supported or needed.

SUMMARY OF DYNAMIC C FEATURES

- Function chaining
- Single Stepping
- Supports embedded assembly code
- Watch Expressions
- Co statements
- Execution Tracing

8. Functions used in Dynamic C

To modify, the pin status using the GUI the following functions are used:

(i) brdInit() : Board Initialization

Calling this function at the beginning of program initializes the system I/O ports and loads all the A/D converter and D/A converter calibration constants from flash memory into SRAM for use by the program.

(ii) digIn(int channel) : Digital Input

Reads the state of a digital input channel. If a configurable I/O channel (DIO00–DIO15) that was configured as a digital output is read by digIn, then the value read will be the state of the output channel.
A run-time error will occur for the following conditions:

- channel out of range.
- brdInit was not executed before executing digIn.

Parameter: channel is the input channel number (0–15 for DIO00–DIO15)
In our example, channel=0x0FE00
i.e channel= 1111111000000000(DIN16-DIN31)

(iii) digOut(int channel, int state) : Digital Output

Sets the state of a configurable I/O channel (DIO00–DIO15) configured as a sinking digital output to a logic 0 or a logic 1. This function only allows control of channels that are configured to be an output by the digOutConfig function. Remember to call the brdInit and the digOutConfig functions before executing this function.

A runtime error will occur for the following conditions:

- channel or state out of range.
- brdInit or digOutConfig was not executed before executing digOut.

Parametrs:

Channel is the output channel number (0–15).State sets a given channel to one of the following output states.

0 = connect the load to GND

1 = put the output in a high-impedance state

(iv) digOutConfig(int configuration) : Digital Output Configuration

Configures any of the 16 configurable I/O channels to be a sinking output. This configuration informationis then used by the digOut function to determine whether a given channel is configured to be an output. If it is not, digOut will prevent the given channel from being used by the digOut function. configuration is a 16-bit parameter for which each bit corresponds to a configurable I/O channel (0–15).

In our example, configuration = 0x540E

0x540E = 0101010000001110 (bits 15–0)

(v) anaInConfig(int ch_pair, int opmode) : Analog Input Configuration

Configures an A/D converter input channel pair for a given mode of operation. This function must be called before accessing the A/D converter chip.

Parametres:

ch_pair are the channel pairs:

- 0 = channels 0 and 1
- 1 = channels 2 and 3
- 2 = channels 4 and 5
- 3 = channels 6 and 7

opmode selects the mode of operation for the channel pair on A/D converter:

- 0 = Single-Ended unipolar (0–10 V)
- 1 = Single-Ended bipolar (±10 V)
- 2 = Differential bipolar (±20 V)
- 3 = 4–20 mA

(vi) anaOutConfig(char configuration, int mode) : Analog Out Configuration

Configures the D/A converter chip for a given output voltage range, 0–10 V or ±10 V, and loads the calibration data for use by the D/A converter API functions. This function must be called before accessing any of the D/A converter channels.

Parameters: configuration sets the output configuration as follows:

- 0 = unipolar operation. (0–10V and 4–20 mA)
- 1 = bipolar operation. (±10V and 4–20 mA)

Mode :

- 0 = asynchronous—an output is updated at the time data are written to the given channel

- 1 = synchronous—all outputs are updated with data previously written when the anaOutStrobe function is executed.

(vii) digHoutConfig(char configuration) :

Configures a high-current output to be either a sinking or a sourcing output. This configuration information is also used to initially set the output to the off state for the given hardware output configuration. The configuration options are described below.
Parameter:
Configuration is a 1-byte parameter where 4 bits are used for the high-current outputs HOUT0–HOUT3.
Bit 3 = high-current output channel HOUT3
Bit 2 = high-current output channel HOUT2
Bit 1 = high-current output channel HOUT1
Bit 0 = high-current output channel HOUT0
(Bits 4–7 are not used)
The high-current outputs can be configured to be sinking or sourcing outputs by setting the corresponding bit to an 0 or 1 : 0 = sinking, 1 = sourcing.

(viii) int serMode(int mode) **-** Sets up the serial communication lines.
User interface to set up BL2600 serial communication lines. Call this function after serXOpen(). Whether you are opening one or multiple serial ports, this function must be executed after executing the last serXOpen function AND before you start using any of the serial ports. This function is non-reentrant.
If Mode 1 is selected, CTS/RTS flow control is exercised using the serCflowcontrolOn and serCflowcontrolOff functions from the RS232.LIB library.
PARAMETER:
mode is the defined serial port configuration.

(ix) void ser485Tx(void) - Enables the RS-485 transmitter. serMode() must be executed before running this function. Transmitted data are echoed back into the receive data buffer. The echoed data could be used to identify when to disable the transmitter by using one of the following methods.
Byte mode—disables the transmitter after the byte that is transmitted is detected in the receive data buffer.
Block data mode—disable the transmitter after the same number of bytes transmitted are detected in the receive data buffer.

(x) void ser485Rx(void) –
Disables the RS-485 transmitter. This puts you in listen mode, which allows you to receive data from the RS-485 interface. serMode() must be executed before running this function.

(xi) int anaInCalib(int channel, int opmode, int gaincode, int value1, float volts1, int value2, float volts2) - Calibrates the response of a given A/D converter channel as a linear function using the two conversion points provided.
PARAMETERS:
Channel : is the analog input channel number (0 to 7) corresponding to AIN0–AIN7.
Opmode : is the mode of operation for the specified channel. Use one of the following macros to set the mode for the channel being configured.
0 = Single-Ended unipolar (0–20 V)
1 = Single-Ended bipolar (±10 V)
2 = Differential bipolar (±20 V)
3 = 4–20 mA
Gaincode : is the gain code of 0 to 7 (gain code of 4 for 4–20 mA operation).
value1 is the first A/D converter value (0–4095).
volts1 is the voltage corresponding to the first A/D converter value.
value2 is the second A/D converter value (0–4095).
volts2 is the voltage corresponding to the second A/D converter value.

9. Conclusion

We have successfully created a GUI to configure the pins present on BL2600. The front end of the GUI has been made in V**B** and the backend involves the generation of Dynamic C code in notepad of the windows operating system. This was accomplished by reading the arguments passed from the GUI into a text file and then reading the port values from this text file through C++ and passing them into the dynamic C code.
Hence the front end is VB, backend is Dynamic C and C++ does the linking of the two.
We have also used RCM3400 and successfully compiled various sample programs such as Controlled, Flashled, Toggleswitch and also programmed RCM3400 to check status and control the LED's using Dynamic C and the related TCP/IP features.