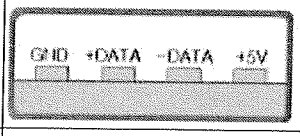
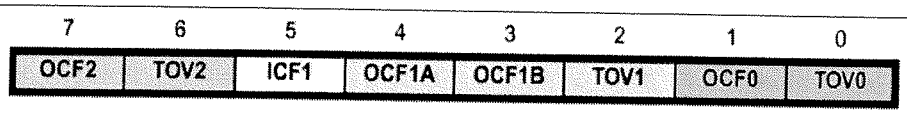
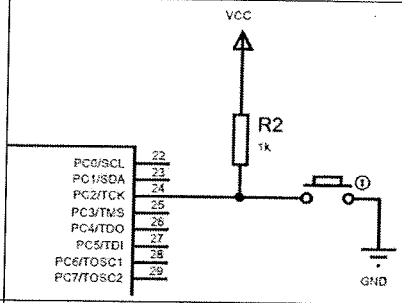
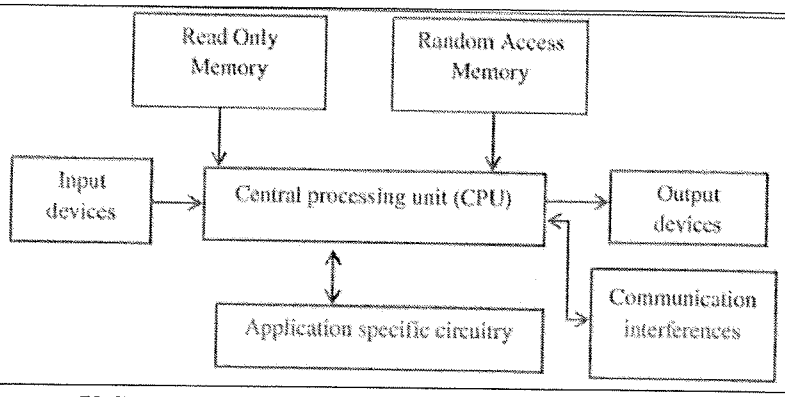
SCORING INDICATORS

COURSE NAME: **EMBEDDED SYSTEMS**
COURSE CODE: – 6201B QID: 2102240184

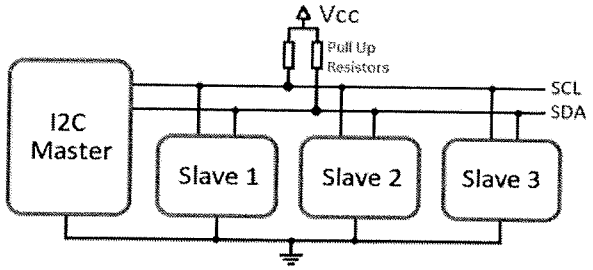
QNo	ScoringIndicators	Split score	Sub Total	Total score
	PARTA			**9**
I.1	Washing machines, Television, Set top Boxes ,Microwave ovens, Dish washer, refrigerators (Any 2)		1	
I.2	A sensor is transducer that is used to generate an input signal to a measurement, instrumentation or control system.		1	
I.3			1	
I.4			1	
I.5	Timer 0 , Timer 2		1	
I.6			1	
I.7	MISO,MOSI,SCK,SS (Any 2)		1	
I.8	The memory location at which the kernel code is located is known as kernel space		1	
I.9	Multiprocessor systems		1	
	PARTB			**24**
II.1	1) Based on Generation: The classification of embedded systems is based on the generation in which they are evolved from its initial version to the latest version. • First Generation • Second Generation • Third Generation • Fourth Generation 2) Based on Complexity & Performance: The embedded systems are classified into three types based on the complexity & performance of the systems.		3	

	• Small Scale Embedded Systems • Medium Scale Embedded Systems • Large scale Embedded Systems			
II.2			3	
II.3	• I2C • SPI • UART • 1-Wire • Parallel bus interface.		3	

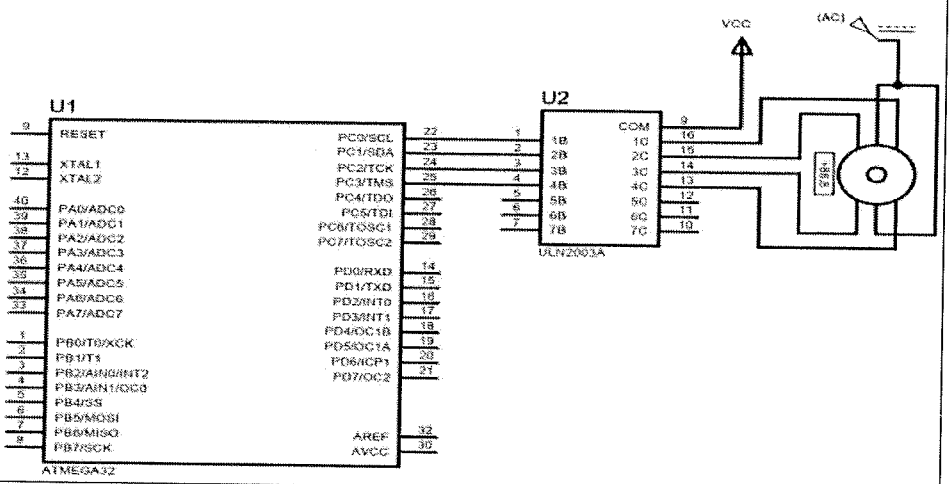
II.4	#include <avr/io.h>
	#include <util/delay.h>
	int main(void)
	{
	DDRB = 0xFF;
	DDRC = 0x00;
	
	while (1)
	{
	// Check the state of bit 3 in Port C
	if (PINC & (1 << 3))
	{
	// Bit 3 is HIGH, send F0H to Port B
	PORTB = 0XF0;
	}
	else
	{
	// Bit 3 is LOW, send 0FH to Port B
	PORTB = 0x0F;
	}
	_delay_ms(1000); // Delay
	}
	return 0;
	}
			3	

II.5	#include <avr/io.h> #include <util/delay.h>		3	

	int main(void)
	{
	// Set all bits of Port B as output
	DDRB = 0xFF;
	while (1)
	{
	// Toggle all bits of Port B
	PORTB ^= 0xFF;
	_delay_ms(3000); // Delay for 3 second
	}
	return 0;
	}			

II.6	 • It is a bus interface connection incorporated into many devices such as sensors, RTC(Real Time Clock) and EEPROM. • I2C ideal for attaching low speed peripherals to a motherboard or embedded system or anywhere that a reliable communication over a short distance is required. • I2C devices use only 2 pins for data transfer. • SCL(Serial Clock): The line carries the clock signal & it synchronize the data transfer. • SDA(Serial Data): The line send and receive data. • SDA and SCL make the I2C a 2 wire interface(TWI). • I2C devices use only 2 bidirectional open drain pins for data communication.		3	

II.7	#include
	int main(void)
	{
	DDRC = 0xFF; // PORT C is output
	DDRD = 0x00; // PORT D is input
	while(1)
	{
	if (PIND & (1<<2) // Checking the status of pin PD2
	{
	PORTC = 0x00;
	} else
	{
	PORTC = 0xFF;
	}
			3	

	} return 0; }			
II.8			3	
II.9	A kernel is a central component of an operating system that acts as an interface between the user applications and the hardware. It contains a set of system libraries & services. For a general purpose OS, the kernel performs following functions: • Process Management • I/O System (Device) Management • File System Management • Memory Management • Time Management		3	
II.10	Multitasking involves the execution switching among different tasks. Determining which task/process is to be executed at a given point of time is known as task scheduling. The kernel service which implements the scheduling algorithm is known as 'Scheduler'. The process of scheduling decision may takes place when a process switches its state to: 1) 'Ready' state from 'Running' state 2) 'Blocked/Wait' state from 'Running' state 3) 'Ready' state from 'Blocked/Waiting' state 4) 'Completed' state		3	
	PARTC			**42**

III

Paramater	General Purpose Computer	Embedded System
Basic	It is a general purpose electronic device used to perform different types of tasks.	It is a specialized computer system that used to perform one or a few specific tasks.
System hardware	A computer typically consists of a CPU, storage unit, and I/O units.	Embedded system are designed with a microcontroller which consists of a CPU, memory unit, and I/O interface on a single IC chip.
Processing power	High processing power.	Relatively low processing power.
Storage capacity	High storage capacity or memory to store data and information on the system.	Less memory capacity as compared to computers.
Size	Generally larger in size.	Smaller in size than computers.
Cost	More expensive than embedded systems.	Less expensive.
Operating system	Computers use a full-featured operating system to run.	Embedded systems use a specialized operating system to run.
Software development tools	For computers, the general purpose development tools can be used to develop computer software.	The development of software for embedded systems requires specialized and expert tools.

7

IV

Universal Asynchronous Receiver Transmitter (UART):

It is an asynchronous form of serial data transmission. It doesn't require a clock signal to synchronize the transmitting end and receiving end for transmission. Instead UART uses start and stop bits with actual data bits, which defines the start and end of data packet. For proper communication, the Transmit line (TX) of the sending device should be connected to the Receive line (RX) of the receiving device. UART works under full duplex communication mode meaning it can transmit and receive data at same time.

RS232 (Recommended Standard 232):

It is a full duplex, wired, asynchronus mode serial communication interface developed by Electronics Industrial Assosiation(EIA). It extend the UART communication signal for external data communication.

Universal Serial Bus (USB):

It is a common interface that allows the connection between external devices and controllers. It connects peripheral devices including digital cameras, mice, keyboards, printers, scanners, media devices, external hard drives, and flash drives. USB protocol sends and receives the data serially between host and external peripheral devices through data signal lines D+ and D-. Apart from two data lines, USB has VCC and Ground signals to power up the device.

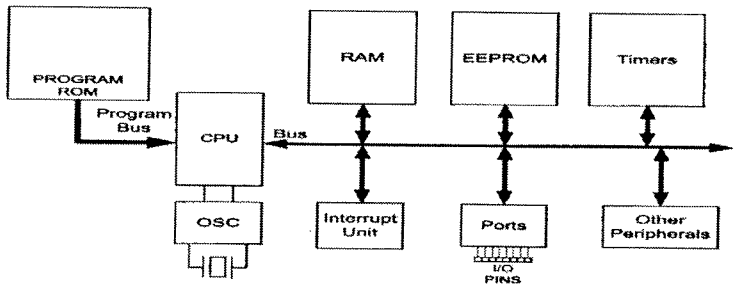
Bluetooth:

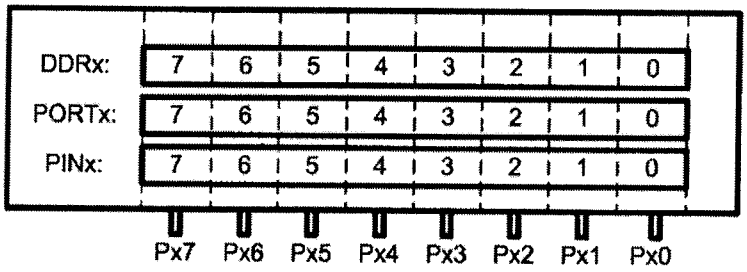
it is a Low-cost short-distance radio communications standard. Bluetooth is wireles and replaces cable technology. It uses radio waves for transmitting and receiving data. It operates at 2.4GHz of radio spectrum.

Wireless Fidelity(Wi-Fi):

It is the popular wireless communication technique for networked communication of devices. Wi-Fi operates at 2.4GHz or 5GHz of radio

7

	spectrum. Wi-Fi supports Internet Protocol (IP) based communication. In an IP based communication each device is identified by an IP address, which is unique to each device on the network. Wi-Fi enabled devices contain a wireless adaptor for transmitting and receiving data in the form of radio signals through an antenna (Wi-Fi Radio).			
V	AVR family: AVRs are generally classified into 4 broad groups: • Classic AVR (AT90SXXXX):This is the original AVR chip which has been replaced by newer chips. Some members of this family are – AT90S2313, AT90S2323, AT90S4433. • Mega AVR (ATMegaXXXX): These are powerful micro controllers with more than 120 instructions and lots of different peripheral capabilities. Some members of this family are – ATMega8, ATMega16, ATMega32, ATMega64, ATMega128. • Tiny AVR (ATTiny XXXX): Micro controller in this group have less instructions and smaller packages when compared to mega family. Some members of this family are – ATTiny13, ATTiny25, ATTiny44, ATTiny84. • Special Purpose AVR: They are made with special capabilities for specific applications. Examples for special capabilities are USB controller, CAN controller, LCD controller, Zigbee, Ethernet controller, FPGA and advanced PWM. Some members of this family are – AT90CAN128, AT90USB128, AT90PWM128.		7	
VI	 I/O ports: AT Mega32 has four ports(Port A,Port B, port C and Port D) having 32 pins. Oscillator: AT Mega32 has an internal oscillator for its clock. By default it is set to operate an internal caliberated oscillator of 1 MHz. Timer/counter: It is used to count an event or to generate time delays between two operations. AT Mega32 consist of two 8 bit(Timer0 & Timer2) and one 16 bit timer/counter(Timer1). Watchdog timer: A watchdog timer is a simple countdown timer which is used to reset a microprocessor after a specific interval of time. System reset is required for preventing failure of the system in a situation of a hardware fault or program error. Interrupt unit: AVR ATmega32 consist of 21 interrupt sources out of which three are external. The remaining are internal interrupts which supports the peripherals like USART, ADC, timer etc. The three external hardware interrupts are on pins PD2, PD3, and PB2 which are referred to as INT0, INT1, and INT2 respectively. Memory: AT Mega32 consist of three different memory sections. • Program ROM: It is used to store program or codes. AT	Fig 3 Exp 4	7	

	Mega32 has 32KB as Program ROM • EEPROM: It is also a non volatile memory used to store data. AT Mega32 has 1 KB of EEPROM. • Data RAM: It is a volatile memory used to store data. AT Mega32 has 2 KB of Data RAM. Other Peripherals: It include: •ADC interface: AT Mega32 has an 8 channel ADC with resolution of 10 bits for Analog to Digital signal conversions. •USART(universal synchronous asynchronous receiver transmitter): USART interface is available for interfacing with external device capable of communicating serially. •SPI(Serial Peripheral Interface): It is used for serial communication between two devices on a common clock source. The data transmission rate of SPI is more than that of USART. •TWI(Two Wire Interface): Can be used to attach low speed peripherals to a motherboard, embedded systems, cell phones or other electronic devices. It is a multi master serial single ended computer bus.			
VII	 Atmega32 has 4 ports(PORTA, PORTB, PORTC, PORTD) having 32 pins assigned to these ports. It is dedicated to special functions such as I/O operations, timer, serial communication, interrupt, ADC etc. To use any port as input or output, it must be programmed. . Each port has three 8-bit I/O registers associated with it. They are designated as PORTx, DDRx & PINx. <u>DDRx Register:</u> It is used for the purpose of making a given port an input or output port. • To make a output port write 1s to DDRx register. • To make a input port write 0s to DDRx register. <u>PINx Register:</u> To read the data present at a pin to CPU, we use the PINx register. <u>PORTx Register:</u> To sent the data present in CPU to a pin, we use the PORTx register.	Fig 3 Exp 4	7	
VIII	In C programming, data types are declarations for variables. This determines the type and size of data associated with variables.		7	

	Data Type / **Size in Bits** / **Data Range/Usage** unsigned char / 8-bit / 0 to 255 char / 8-bit / –128 to +127 unsigned int / 16-bit / 0 to 65,535 int / 16-bit / –32,768 to +32,767 unsigned long / 32-bit / 0 to 4,294,967,295 long / 32-bit / –2,147,483,648 to +2,147,483,648 float / 32-bit / ±1.175e-38 to ±3.402e38 double / 32-bit / ±1.175e-38 to ±3.402e38 1) Unsigned char: It is an 8-bit data type that takes the value in the range of 0-255(00-FFH). It is one of the most widely used data types for AVR like setting counter values. The C-compiler by default use the signed char unless the keyword 'unsigned' is specified. 2) Signed char: It is an 8-bit data type that uses the Most Significant Bit(MSB) i.e D7 bit to represent +ve or -ve value. As a result only 7 bits are available for use, giving the rang -128 to +127. 3) Unsigned int: It is an 16-bit data type that takes the value in the range of 0-65,535(0000- FFFFH). It is used to define 16-bit variable such as memory address, set counter values more than 256. The C-compiler by default use the signed int unless the keyword 'unsigned' is specified. Since it is a 2-byte data type, int is used only in situations that demands, because the use of int data type will result in the creation of larger hex files, slower execution and more memory usage. 4) Signed int: It is an 16-bit data type that uses the Most Significant Bit(MSB) i.e D15 bit to represent +ve or -ve value. As a result only 15 bits are available for use, giving the rang -32,768 to +32,767. 5) 5) Other data types: • If we want values greater than 16-bit use long data types. • If we want deal with fractional numbers use float & double data types.			
IX	#define F_CPU 1000000UL //1 MHz clock #include <avr/io.h> #include <util/delay.h> int main(void) {		7	

```
                DDRA = 0xFF;                //LCD data pins connected to Port A
                DDRB = 0xFF;                //LCD ctrl pins connected to Port B
                PORTB &= ~(1<<2);           //EN = 0
                _delay_us(2000);
                lcdCommand(0x38);           //2 line 5X7 matrix
                lcdCommand(0x0E);           //display on cursor on
                lcdCommand(0x01);           //clear lcd
                _delay_us(2000);
                lcdCommand(0x06);           //shift cursor right
                lcdCommand(0x80);
                lcd_print("Hello");
                lcdCommand(0xC0);
                lcd_print("World");

        while(1);
        return 0;
        }

void lcdCommand(unsigned char cmnd)
        {
                PORTA = cmnd;
                PORTB &= ~(1<<0);           //RS = 0 for command
                PORTB &= ~(1<<1);           //RW = 0 for write
                PORTB |= (1<<2);    /       /EN=1 for H to L pulse
                _delay_us(1);
                PORTB &= ~(1<<2);           //EN=0 for H to L pulse
                _delay_us(100);
        }

void lcdData(unsigned char data)
        {
                PORTA = data;
                PORTB |= (1<<0);            //RS = 1 for data
                PORTB &= ~(1<<1);           //RW = 0 for write
                PORTB |= (1<<2);            //EN=1 for H to L pulse
                _delay_us(1);
                PORTB &= ~(1<<2);           //EN=0 for H to L pulse
                _delay_us(100);

}
void lcd_print(char *str)
        {
                unsigned char i = 0;
                while(str[i] != 0)
                {
                        lcdData(str[i]);
                        i++;
                }
```

X	#include #include int main(void) {		7	

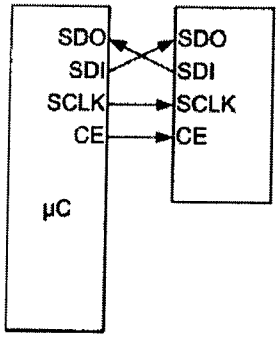
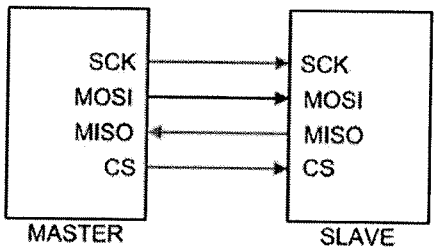
```
    DDRC = 0x01; //Makes PC 0 output pin
    PORTC = 0x00;
      while(1)
       {
         //Rotate Motor to 0 degree by applying 1 µs pulse
          PORTC = 0x01;
           delay_us(1000);
          PORTC = 0x00; _
          delay_ms(2000); //Rotate Motor to 90 degree by applying 1.5µs
pulse
          PORTC = 0x01; _
          delay_us(1500);
          PORTC = 0x00; _
          delay_ms(2000); //Rotate Motor to 180 degree by applying 2µs
pulse
          PORTC = 0x01; _
          delay_us(2000);
          PORTC = 0x00; _
          delay_ms(2000);
       }
      return 0;
}
```

XI

ADC(Analog-to-Digital Conversion) interfacing:

- Digital computers use binary values, but in physical world everything is analog.
- The physical quantity(temperature, humidity, pressure..) is converted to electrical signal(voltage ,current..)using sensors(transducer) .
- Sensors produce an output voltage or current which is analog.
- So we need an ADC to translate analog to digital numbers.
- ATmega32 has an on-chip ADC:
  - 10 bit ADC
  - 8 analog input channel(ADC0,ADC1,…..ADC7), 7 differential input channels, 2 differential input channels with optional gain of 10x and 200x.
  - The converted output binary data is held by two special function registers: ADCL and ADCH.
  - ADCH:ADCL registers can hold 16 bits, but ADC output is 10 bits wide, 6 bits of the 16 are unused. We have the option of making either upper 6 bits or lower 6 bits unused
  - We have 3 options for Vref . Vref can can be connected to Avcc(Analog Vcc), internal 2.56V reference or external AREF pin.
  - Conversion time is dictated by the crystal frequency connected to XTAL pins(Fosc) and ADPSC 0:2 bits.
  - 5 registers associated with ADC
    - ADCH(high data)
    - ADCL(low data)

7

	• ADCSRA(ADC Control & Status Register) • ADMUX (ADC Multiplexer) • SPIOR(Special function I/O register) ADC start conversion: The ADSC bit of ADCSRA register is the start conversion input. Analog to Digital conversion time: By using ADSP2-ADSP0 bits of ADCSRA register we can select the A/D conversion time.			
XII	Serial Peripheral Interface (SPI): • It is a bus interface connection protocol originally started by Motorola Corp(Freescale). • SPI is a synchronous serial interface in which data in an 8-bits can be shifted in and/or out one bit at a time. • It can be used to communicate with a serial peripheral device(flash, EEPROM, sensors) or with another microcontroller with an SPI interface • It uses four pins for communication. ▪ SDI (Serial Data Input) ▪ SDO (Serial Data Output) ▪ SCLK (Serial Clock) ▪ CE(Chip Enable)/CS (Chip Select) • It has two pins for data transfer called SDI (Serial Data Input) and SDO (Serial Data Output). SCLK (Serial Clock) pin is used to synchronize data transfer and Master provides this clock. CS (Chip Select) pin is used by the master to select the slave device.   • SPI peripherals in AVR can operate in Master or Slave modes. If master mode is selected, then it takes care of generating the clock signal and starting the transfer. The slave only waits for the CS line to pull down to get ready for transfer. • ATmega32 has an inbuilt SPI module. It can act as a master and slave SPI device. SPI communication pins in AVR ATmega are: ▪ MISO(Master In Slave Out): The Master receives data and the slave transmits data through this pin. ▪ MOSI(Master Out Slave In): The Master transmits data and the slave receives data through this pin. ▪ SCK(Synchronization Clock): The Master generates this clock for the		7	

	communication, which is used by the slave. The only master can initiate a serial clock. ▪ SS(Slave Select): Master can select slaves through this pin.			
XIII	Selection criteria for RTOS: The factors include the functional & non-functional requirements for selection an RTOS. Functional Requirements: These are requirements that specify the desired functionality or operations of a RTOS. It includes: • Processor Support: It is essential to ensure that the processor should support the OS under consideration. • Memory Requirement: Since embedded system is memory constrained, it is essential to have a minimal ROM & RAM requirements for an OS under consideration. • Real time capabilities: It is essential to analyse the real time capabilities of an OS under consideration, as task/process scheduling policies play an important role in real time behaviour • Kernel & Interrupt Latency: The kernel of the OS may disable interrupts while executing certain services & it may lead to interrupt latency. This latency should be minimal for a high response system. • Inter Process Communication & Task Synchronization: This is a kernel dependent parameter & certain kernel implement policies for avoiding issues in resources sharing. • Modularization Support: It allows the developers to choose essential modules & recompile OS image for functioning. • Support for Networking & Communication: This include driver support for networking & communication interfaces. • Development Language Support: Certain OS requires run timr libraries for application written in languages like Java, C#, .Net etc. Non Functional Requirements: They are not related to the software's functional aspect. It concentrates on the expectation and experience of the user. • Custom developed or off the shelf: Custom-developed OS are specifically designed and developed to meet an embedded system's specific needs, while off-the-shelf OS are pre-built OS that can be purchased and used immediately. • Cost: It include developing cost, buying cost, maintenance cost etc. • Development and debugging tools availability: It is a critical factor for selecting an OS, as some OS may be superior in performance, but availability of development & debugging tools are limited. • Ease of use: It specify how easy to use a commercial OS. • After sales: It includes bug or error fixing, patch updation, online support are important factor for selecting an OS.		7	
XIV	1. FreeRTOS (Amazon)		7	

	2. Zephyr (Linux Foundation) 3. MQX (Philips NXP / Freescale) 4. Keil RTX (ARM) 5. µC/OS (Micrium) 6. LynxOS (Lynx Software Technologies) 7. Integrity (Green Hills Software) 8. embOS (SEGGER) 9. ThreadX (Microsoft Express Logic) 10. Neutrino (BlackBerry)			