**Questions:**

**Embedded C**

1. How to measure the size of any variable and data type without using “sizeof” operator?

Note: replace sizeof() with my\_sizeof() in the following example:

1. sizeof(long long int);
2. short int a; sizeof(a);

**Ans:**

a)

#define my\_sizeof (T) (((T \*)0)+1)  
void main()  
{  
 printf(“%d”, my\_sizeof (long long int));  
}

b)

#define my\_sizeof (x) ((char \*)(&x+1) – (char \*)&x)  
void main()  
{  
 short int x;  
 printf(“%d”, my\_sizeof (x));  
}

1. Write a function like Macro to convert Little Endian to Big Endian.

**Ans:**

#define REVERSE\_LONG(n) ((unsigned long) (((n & 0xFF) << 24) | \

((n & 0xFF00) << 8) | \

((n & 0xFF0000) >> 8) | \

((n & 0xFF000000) >> 24)))

1. Write the prototype of a function pointer and give an example.

**Ans:**

return\_type (\* func\_pointer\_name) (datatype\_arg\_1, datatype\_arg\_1, ...);

void \*(\*fp) (int \*, int \*);

1. Write APIs to convert ASCII to Integer and Integer to ASCII without using atoi or itoa functions.

**Ans:**

int myAtoi(char\* str)

{

// Initialize result

int res = 0;

// Initialize sign as positive

int sign = 1;

// Initialize index of first digit

int i = 0;

// If number is negative,

// then update sign

if (str[0] == '-') {

sign = -1;

// Also update index of first digit

i++;

}

// Iterate through all digits

// and update the result

for (; str[i] != '\0'; ++i)

res = res \* 10 + str[i] - '0';

// Return result with sign

return sign \* res;

}

// Implementation of itoa()

char\* itoa(int num, char\* str, int base)

{

int i = 0;

char isNegative = 0;

/\* Handle 0 explicitly, otherwise empty string is printed for 0 \*/

if (num == 0)

{

str[i++] = '0';

str[i] = '\0';

return str;

}

// In standard itoa(), negative numbers are handled only with

// base 10. Otherwise numbers are considered unsigned.

if (num < 0 && base == 10)

{

isNegative = 1;

num = -num;

}

// Process individual digits

while (num != 0)

{

int rem = num % base;

str[i++] = (rem > 9)? (rem-10) + 'a' : rem + '0';

num = num/base;

}

// If number is negative, append '-'

if (isNegative)

str[i++] = '-';

str[i] = '\0'; // Append string terminator

// Reverse the string

reverse(str, i);

return str;

}

void reverse(char str[], int length)

{

int start = 0;

int end = length -1;

while (start < end)

{

swap(\*(str+start), \*(str+end));

start++;

end--;

}

}

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

1. What is the difference between Null Pointer, Void Pointer and Wild Pointer?

**Ans:**

Void pointer is a specific pointer type – void \* – a pointer that points to some data location in storage, which doesn’t have any specific type. Void refers to the type. Basically the type of data that it points to is can be any. If we assign address of char data type to void pointer it will become char Pointer, if int data type then int pointer and so on. Any pointer type is convertible to a void pointer hence it can point to any value.

NULL Pointer is a pointer which is pointing to nothing. In case, if we don’t have address to be assigned to a pointer, then we can simply use NULL.

A pointer that has not been initialized to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

1. Write the function prototypes of malloc, calloc, realloc and free and Explain how free is able to deallocate memory without mentioning the size of the data?

ptr = (cast-type\*) malloc(byte-size)

ptr = (cast-type\*)calloc(n, element-size);

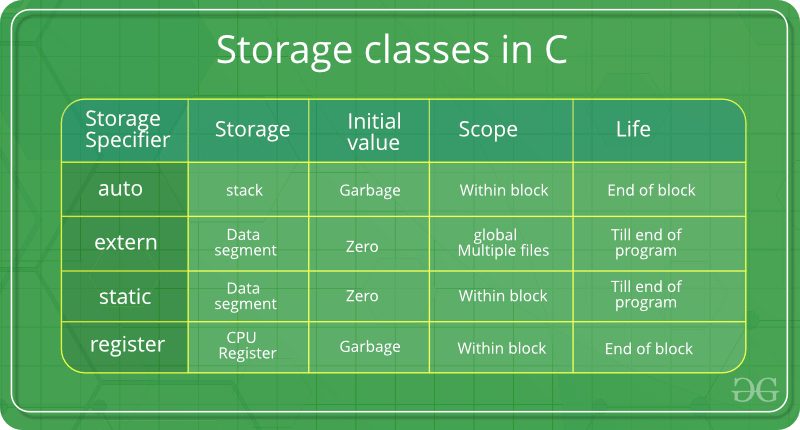
ptr = realloc(ptr, newSize);

free(ptr);

Ans:

When memory allocation is done, the actual heap space allocated is one word larger than the requested memory. The extra word is used to store the size of the allocation and is later used by free( ).

1. Mention the default values, memory location, life time and scope for the following storage classes:
   1. Auto
   2. Register
   3. Local static
   4. Global Static
   5. Extern



**Global static – Scope – Within File**

**Local static – Scope – Within Block**

1. Find the output of the following Program:

**int** main()

{

**int** a[5] = {1,2,3,4,5};

**int** i;

**for** (i = 0; i < 5; i++)

**if** ((**char**)a[i] == '5')

printf("%d\n", a[i]);

**else**

printf("FAIL\n");

**return** 0;

}

Ans:

FAIL  
FAIL  
FAIL  
FAIL  
FAIL

1. Find the output of the following Program:

**#define ZERO (0u)**

**int** main()

{

int i1;

**float** f1 = 0.1;

**if** (f1 == 0.1)

printf("equal\n");

**else**

printf("not equal\n");

for(i1 = 3; i1 >= ZERO ; i1--)

{

printf(“%d\t”,i1);

}

**return** 0;

}

Ans:

not equal

infinite loop

**Communication Protocols**

1. Write the Frame Format of the following:
   1. UART
   2. I2C 7 bit Addressing Mode
   3. I2C 10 bit Addressing Mode
   4. CAN Standard Data Frame
   5. CAN Extended Data Frame

Ans:

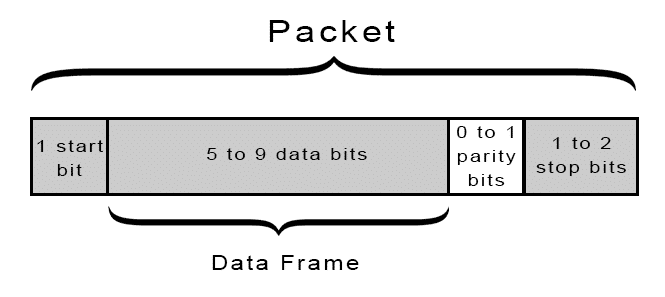


Figure 1 UART frame

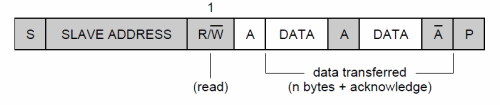


Figure 2 I2C 7 bit Write Frame Format

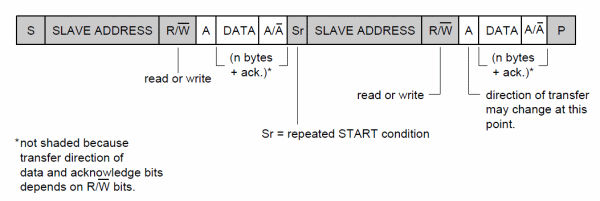


Figure 3 I2C 7 bit read frame format

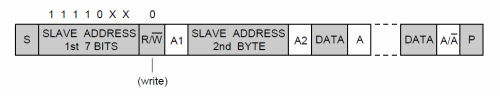


Figure 4 I2C 10 bit Address Write Frame

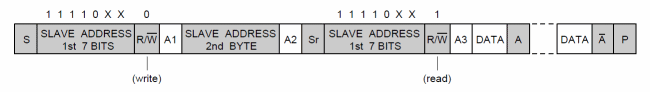


Figure 5 I2C 10 bit Address Read Frame

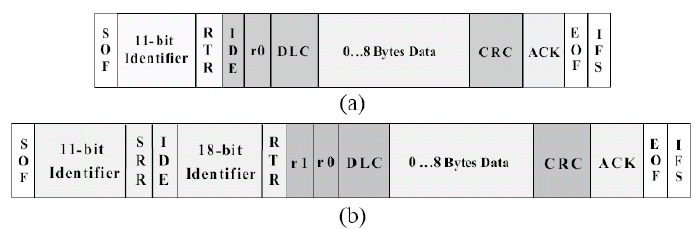


Figure 6 : CAN Standard and Extended Data Frame

1. List down the standard speeds of the following communication protocols:
   1. UART – 9600 bps, 115200 bps, etc.
   2. I2C – 100kbps, 400kbps, 1Mbps, 3.4Mbps
   3. CAN – 250kbps, 500kbps, 1Mbps etc.
2. Mention the pin outs of the following communication Protocols:
   1. UART – RX, TX, GND
   2. I2C – SCL, SDA, GND
   3. SPI – MISO, MOSI, SCK, CS, GND
   4. CAN – CANH, CANL
3. Mention the transmission mode (Full Duplex or Half Duplex) for the following communication protocols:
   1. UART – Full Duplex
   2. I2C – Half Duplex
   3. SPI – Full Duplex
   4. CAN – Half Duplex
4. If there are two I2C slaves one supports 7 bit addressing mode and another supports 10 bit addressing mode. Will you be able to communicate with both using a single I2C master, if yes, how?

Yes, because the i2c 10 bit address frame is designed in such a way that it master can communicate with both slaves at the same time. For 10 bit addressing the address frame is divided into two 8 bits frames. The first frame consists of 1 1 1 1 0 A9 A8 R/W and next frame consists of A7 – A0.

1. If there are two CAN nodes one broadcasts standard data frames and another broadcasts extended data frames. Will you be able to read both data frames using a single CAN node, if yes, how?

Yes, because the CAN message frame is designed in such a way that it can read both extended and standard IDs, However Extended ID 7DF is different from Standard ID 7DF. But we will be able to read both in the same bus using the IDE bit in the data frame, if speed of communication is same.