

## CN LAB

### LAB 10 (prog 2)

#### Write a program for error detecting code using CRC-CCITT (16-bits)

Program :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>

// Compute CRC-16/CCITT bitwise on a bit-string of '0'/'1'
uint16_t bitwise_crc16_ccitt_bits(const char *messageBits) {
    const uint16_t polynomial = 0x1021;
    uint16_t crc = 0xFFFF; // initial value
    size_t len = strlen(messageBits);

    for (size_t i = 0; i < len; i++) {
        uint8_t bit = (messageBits[i] == '1') ? 1u : 0u;
        crc ^= (uint16_t)(bit << 15);
        for (int b = 0; b < 1; b++) {
            if (crc & 0x8000) {
                crc = (crc << 1) ^ polynomial;
            } else {
                crc <<= 1;
            }
        }
    }
    return crc;
}
```

```

int main(void) {
    const char *dataBits = "101101";
    int crcBits = 16;

    printf("Original data bits      : %s\n", dataBits);

    size_t dataLen = strlen(dataBits);
    size_t augLen = dataLen + crcBits;
    char *augBits = malloc(augLen + 1);
    strcpy(augBits, dataBits);
    for (int i = 0; i < crcBits; i++) {
        augBits[dataLen + i] = '0';
    }
    augBits[augLen] = '\0';
    printf("Augmented data bits     : %s\n", augBits);

    uint16_t remainder = bitwise_crc16_ccitt_bits(augBits);
    printf("Remainder (hex)         : 0x%04X\n", remainder);

    // Convert remainder to binary string
    char *remBin = malloc(crcBits + 1);
    remBin[crcBits] = '\0';
    for (int i = crcBits - 1; i >= 0; i--) {
        remBin[i] = ( (remainder & 1) ? '1' : '0' );
        remainder >>= 1;
    }
    printf("Remainder (binary)       : %s\n", remBin);

    // Form codeword
    char *codeword = malloc(dataLen + crcBits + 1);

```

```

strcpy(codeword, dataBits);

strcat(codeword, remBin);

printf("Transmitted codeword bits : %s\n", codeword);

// Simulate error: flip bit at some position

int flipPos = 2; // as example, flip the 3rd bit (0-based index)

printf("\n*** Simulating error: flipping bit at position %d ***\n", flipPos);

if (codeword[flipPos] == '0') codeword[flipPos] = '1';

else codeword[flipPos] = '0';

printf("Received codeword bits : %s\n", codeword);

// Receiver side: compute CRC on received codeword

uint16_t checkRemainder = bitwise_crc16_ccitt_bits(codeword);

printf("Receiver remainder (hex) : 0x%04X\n", checkRemainder);

if (checkRemainder == 0) {

    printf("No error detected.\n");

} else {

    printf("** Error detected! **\n");

}

free(augBits);

free(remBin);

free(codeword);

return 0;

}

```

OUTPUT:

```
ERROR!
Original data bits      : 101101
Augmented data bits    : 101101000000000000000000
Remainder (hex)        : 0xF7B0
Remainder (binary)     : 111101110110000
Transmitted codeword bits : 101101111011110110000

*** Simulating error: flipping bit at position 2 ***
Received codeword bits   : 100101111011110110000
Receiver remainder (hex)  : 0x739C
** Error detected! **
```