**Resoluciones parcial 1**

Ejercicio A1:

unsigned char set(unsigned char datos, short linea){

char mask = 0x01;

mask = mask << linea;

return datos | mask;

}

unsigned char set(unsigned char datos, short linea){

char mask = 0xFE;

mask = mask << linea;

return datos & mask;

}

Ejercicio A2:

int get\_band(unsigned char control){

int mask = 0x1;

control = control >> 6;

control = control & mask;

if(control == 1){

cout << “Banda seleccionada AM”;

}else{

cout << “Banda seleccionada FM”;

}

}

unisgned char set\_band(unsigned char control, band\_t nueva\_banda){

if(nueva\_banda == 1){

mask = 0x01;

mask = mask << 6;

control = control | mask;

return control;

}else{

mask = 0xFE;

mask = mask << 6;

control = control & mask;

return control;

}

}

Ejercicio A3:

Igual al B y D del ejercicio 7 de la guia 1

Ejercicio A4:

int get\_synthesizer\_divider(unsigned char puerto){

int mask = 0b00111110;

puerto = puerto & mask;

puerto = puerto >> 1;

return puerto+1;

}

void set\_synthesizer\_divider(int fdiv, unsigned char control){

int SYN = control & 0b00000001;

int AFT = control & 0b10000000;

int BAND = control & 0b01000000;

fdiv = fdiv-1;

fdiv = fdiv << 1;

control = AFT + BAND +fdiv +SYN;

}

**/\* Funciones de respaldo \*/**

bool getBit(char puerto, int bitPos){ //Funcion para leer estado y tener el bit

char mask = 0x1;

puerto = puerto >> bitPos;

puerto = puerto & mask;

return puerto;

}

char bit\_HI(char puerto, int bitPos){ //Funcion High

char mask = 0x1;

mask = mask << bitPos;

return puerto|mask;

}

char bit\_LO(char puerto, int bitPos){ //Funcion Low

char mask = 0xFE;

mask = mask << bitPos;

return puerto&mask;

}

Ejercicio B1:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** |
| **Registro A** |  |  |  |  |  |  |  |  |
| **Registro B** |  | M1 | F1 | R1 | S4 | S3 | S2 | S1 |
| **Registro C** | C4 | C3 | C2 | C1 | P1 |  |  |  |

char R1 = leerRegistro(1); //Registro R1 es puerto B

char R2 = leerRegistro(2); //Registro R2 es puerto C

int S1 = getBit(R1, 0);

int S2 = getBit(R1, 1);

int S3 = getBit(R1, 2);

int S4 = getBit(R1, 3);

int R1 = getBit(R1, 4);

int F1 = getBit(R1, 5);

int M1 = getBit(R1, 6);

if( S1 == 1 || M1 == 1){

R2 = bit\_HI(R2, 4);

}else{

R2 = bit\_LO(R2, 4);

}

if( S2 == 1 || F1 == 1 || R1 = 1){

R2 = bit\_HI(R2, 5);

}else{

R2 = bit\_LO(R2, 5);

}

if( S3 == 1 || M1 == 1){

R2 = bit\_HI(R2, 6);

}else{

R2 = bit\_LO(R2, 6);

}

if( S4 == 1 || R1 == 1){

R2 = bit\_HI(R2, 7);

}else{

R2 = bit\_LO(R2, 7);

}

if( R1 == 1){

R2 = bit\_HI(R2, 3);

}else{

R2 = bit\_LO(R2, 3);

}

escribirRegistro( 2, R2);

Ejercicio B2:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** |
| **Registro A** |  |  |  | SL | SB4 | SB3 | SB2 | SB1 |
| **Registro B** |  |  |  |  |  |  |  |  |
| **Registro C** |  | SE2 | SE1 | LED\_4 | LED\_3 | LED\_2 | LED\_1 | LED\_A |

char R0 = leerRegistro(0); //Registro R0 es puerto A

char R2 = leerRegistro(2); //Registro R2 es puerto C

int SB1 = getBit(R0, 0);

int SB2 = getBit(R0, 1);

int SB3 = getBit(R0, 2);

int SB4 = getBit(R0, 3);

int SL = getBit(R0, 4);

if(SL == 1){

R2 = bit\_HI(R2, 0);

R2 = bit\_LO(R2, 5);

if(SB4 == 0){

R2 = bit\_HI(R2, 6);

}else{

R2 = bit\_LO(R2, 6);

}else{

R2 = bit\_HI(R2, 5);

if(SB1 == 1){

R2 = bit\_LO(R2, 5);

R2 = bit\_HI(R2, 6);

}else{

R2 = bit\_HI(R2, 5);

R2 = bit\_LO(R2, 6);

}

}

escribirRegistro( 2, R2);

Ejercicio B3:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **B7** | **B6** | **B5** | **B4** | **B3** | **B2** | **B1** | **B0** |
| **Registro A** |  | EV | Bprincipal | Bauxiliar | Bp |  |  |  |
| **Registro B** |  |  |  |  |  | M3 | M2 | M1 |
| **Registro C** |  |  |  |  |  |  |  |  |

char R0 = leerRegistro(0); //Registro R0 es puerto A

char R1 = leerRegistro(1); //Registro R2 es puerto B

int M1 = getBit(R1, 0);

int M2 = getBit(R1, 1);

int M3 = getBit(R1, 2);

int Bp = getBit(R0, 3);

int Bauxiliar = getBit(R0, 4);

int Bprincipal = getBit(R0, 5);

int EV = getBit(R0, 6);

if(M1 == 0){

R0 = bit\_HI(R0, 5);

if(Bprincipal == 1){

R0 = bit\_LO(R0, 4);

}

}else{

R0 = bit\_LO(R0, 4);

R0 = bit\_LO(R0, 5);

}

if(M2 == 0 && Bprincipal == 0 && Bauxiliar == 0){

R0 = bit\_HI(R0, 3);

}else{

R0 = bit\_LO(R0, 3);

}

if(M3 == 1){

R0 = bit\_HI(R0, 6);

}else{

R0 = bit\_LO(R0, 6);

}

escribirRegistro( 0, R0);