```
.include "M328PDEF.inc" // Incluye libreria del ATMEGA328P
.equ T0VALUE = 131

.equ MODES = 5//Modos que puede tener la programacion
.def MODE = R20//Registro para el modo actual
.equ MAX_USEC = 10// Maximo de unidades
.equ MIN_UNDER = -1// Valor minimo para que haga underflow
.equ MAX_DSEC = 6//Maximo valor de decenas
.equ MAX_UHR = 10//Maximo de horas en unidades
.equ MAX_DHR = 24//Maximo total de horas
.def MAXDHR = R14//Registro para cantidad de horas
.def COUNTER = R21//Contador para el timer
.def ACTION = R22//Accion actual para que boton fue presionado
.def COUNTERHR = R13//Registro para igualar el maximo de horas
.def COUNTERHRA = R12//Registro " " " en alarma
.def COUNTERDIA = R2//Registro comparar dias
.def COUNTERMES = R5//Registro para llevar en cuenta los meses
.equ MODESLR = 2//Switch para cambio entre display izquierdos o derechos
.def MODELR = R19//Llevar registro de izq/der
.def ACTIONLR = R24//Lleva registro de que accion realizar
.def COUNTER2 = R18//Contador para poder llegar a minutos
.def MINUNDER = R3//Registro para comparar el valor minimo para hacer underflow
.equ MIN_UNDERM = 0//Valor para hacer underflow
.def MINUNDERM = R4//Registro para hacer underflow
```

Se le colocan nombres a todos los registro que se vayan a utilizar con su uso en los comentarios, así como nombras valores límite para poder modificarlos rápidamente, como los modos o las unidades máximas, así como el tiempo del reloj.

```
/**********************Registro que se guardan en la RAM***********************************************/
.dseg
.org SRAM_START
/***Minutos***/
USEC: .byte 1
DSEC: .byte 1
/***Horas***/
UHR: .byte 1
DHR: .byte 1
/***Dias***/
UDIA: .byte 1
DDIA: .byte 1
/***Meses***/
UMES: .byte 1
DMES: .byte 1
PMES: .byte 1
/***Alarma***/
USECA: .byte 1
DSECA: .byte 1
UHRA: .byte 1
DHRA: .byte 1
```

Aquí se coloca todos los registros que se quieren guardar en la RAM, con valores de 1 byte.

```
.cseg
.org 0x0000
    JMP START
//Interrupcion por presionar un boton
.org PCI1addr
    JMP PCINT_ISR
//Ir a interrupcion por Timer
.org OVF0addr
    JMP RUTINA_DE_TIMER0_OV

NUM7: .DB 0x7D, 0x50, 0x6E, 0x76, 0x53, 0x37, 0x3F, 0x70, 0x7F, 0x73, 0x77, 0x0A// Tabla para los valores del display
LIMTMES: .DB 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 2, 4// Tabla limite de meses
/*************************************/
```

Se comienza la sección de código donde tenemos las direcciones a las que saltaremos luego de que hayan interrupciones de botón o del Timer0.

Luego esta la creación de la tabla para los valores del dislplay, y la tabla de los días máximos para cada mes.

```
START:
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16

/*************************************/
// Configuracion MCU

SETUP:
    CLI// Desactivar las interrupciones

    //Configuración de Interrupciones

    //Habilitar interrupciones botontes en puerto C
    LDI R16, (1 << PCINT8) | (1 << PCINT9) |(1 << PCINT10) | (1 << PCINT11)
    STS PCMSK1, R16
    LDI R16, (1 << PCIE1)
    STS PCICR, R16

    // Configurar Prescaler "Principal"
    LDI R16, (1 << CLKPCE)
    STS CLKPR, R16 // Habilitar cambio de PRESCALER
    LDI R16, 0b00000100
    STS CLKPR, R16 // Configurar Prescaler a 16 F_cpu = 1MHz

    //Configurar el tipo de reloj, prescaler 8
    LDI R16, (1<<CS01)
    OUT TCCR0B, R16
    LDI R16, T0VALUE
    OUT TCNT0, R16
```

Primero se desactivan las interrupciones en el SETUP, para que no interrupman el proceso. después se da la configuración de los valores en la pila. Luego en el SETUP, se comienza habilitando la capacidad para interrupciones para los cuatro botones en el puerto C, así como la activación de la interrupción como tal. Luego modificamos el prescaler principal de 16MHz a 1MHz. Y en la configuración del tipo de reloj se carga uno a CS01 para tener un prescaler de 8 y comenzar con T0VALUE, que es uno de las

variables a las que damos valor en la primera imagen para que sea más fácil de modificar.

```
// Habilitar interrupciones del TOV0
LDI R16, (1 <<TOIE0)
STS TIMSK0, R16

// Configurar puertos (DDRx, PORTx, PINx)
//Configurar puerto C como entradas para los botones
LDI R16, 0xF0
OUT DDRC, R16 // Configuramos puerto C como entradas
LDI R16, 0x0F
OUT PORTC, R16 // Pullups en primeros cuatro bits, luego las salidas apagadas

//Configurar puerto D como salidas para los botones
LDI R16, 0xFF
OUT DDRD, R16 // Configuramos puerto D como SALIDAS
LDI R16, 0x00
OUT PORTD, R16 // APAGADOS

//Configurar puerto B como salidas para los botones
LDI R16, 0xFF
OUT DDRB, R16 // Configuramos puerto B como salidas
LDI R16, 0x00
OUT PORTB, R16 // APAGADOS
```

Se habilita las interrupciones para el Timer0. Luego se colocan los valores para los pines del nano. En el puerto C se coloco la parta menos significante como entradas con pullups, y la otra parte como salidas con valores iniciales en 0. En los puertos D y B se colocan todos los valores como salidas con valores iniciales apagados.

```
//Colocar valores iniciales a variables
CLR COUNTER
LDI R16, 0x00
STS USEC, R16
STS DSEC, R16
STS UHR, R16
STS DHR, R16
STS USECA, R16
STS DSECA, R16
STS DDIA, R16
STS DMES, R16
STS PMES, R16
STS DHRA, R16
MOV COUNTERDIA, R16
LDI R16, 0x01
STS UDIA, R16
LDI R16, 0x01
STS UMES, R16
LDI MODE, 0x00
LDI MODELR, 0x00
LDI ACTIONLR, 0x01
CLR ACTION
CLR COUNTERHR
LDI R16, 1
MOV COUNTERHRA, R16
LDI R16, 0x18//Maximo de horas = 24
MOV MAXDHR, R16
LDI R16, 0x0C
MOV COUNTERMES, R16
LDI R16, -1
MOV MINUNDER, R16
LDI R16, 0
MOV MINUNDERM, R16
LDI R16, 0x01
STS UHRA, R16

SEI// Activar las interrupciones
```

Aquí se dan los valores iniciales a las variables para que no inicien en valores al azar al cargar el código, la mayoría se coloca en 0, pero en casos como ACTIONLR, se inicia en 1 porque solo cambia entre dos estados, o el registro que lleva el máximo de horas con un valor inicial de 0C en hexadecimal. Por último, se vuelven a habilitar las interrupciones.

```
MAIN_LOOP:
    CALL REVISAR_ALARMA//Verificar si los valores del reloj y la alarma son iguales
    CPI MODE, 0
    BREQ RELOJ//Ir al reloj
    CPI MODE, 1
    BREQ FECHA //Ir a la fecha
    CPI MODE, 2
    BREQ GO_CONFIG_RELOJ//Ir a configurar reloj
    CPI MODE, 3
    BREQ GO_CONFIG_FECHA//Ir a configurar fecha
    CPI MODE, 4
    BREQ GO_CONFIG_ALARMA//Ir a configurar alarma
    RJMP MAIN_LOOP

GO_CONFIG_RELOJ:
    RJMP CONFIG_RELOJ

GO_CONFIG_FECHA:
    RJMP CONFIG_FECHA

GO_CONFIG_ALARMA:
    RJMP CONFIG_ALARMA
```

Aquí es donde comienza el programa, primero se llama a la función para revisar alarma para verificar si el buzzer debiese de sonar. Luego de revisar se verifica el modo actual en el que se encuentra y cambia dependiendo del registro MODE, si es 0 irá al reloj, 1 a mostrar la fecha, 2 a configurar el reloj, 3 a configurar la fecha y 4 para modificar la alarma. Todas las funciones con GO son saltos a funciones a las que las comparaciones con BREQ ya no pueden llegar.

```
//MODOS RELOJ
RELOJ:
    SBI PORTC, PC4//Activar solo un LED y desactivando la otra
    CBI PORTC, PC5
    SBRC COUNTER, 0//Ciclos para el multiplexeado de los display
    CALL SHOW_USEC
    SBRS COUNTER, 0
    CALL SHOW_DSEC
    SBRC COUNTER, 1
    CALL SHOW_UHR
    SBRS COUNTER, 1
    CALL SHOW_DHR
    SBRS COUNTER2, 1//Cada medio segundo cambiar el estado de las led para los segundos
    CALL CAMBIAR_LED
    RJMP SALIDA
```

El primer modo, reloj, enciende la LED amarilla y apaga la Azul, luego cicla en multiplexado dependiendo de los valores actuales de los bits 0 y 1 del COUNTER, para así mostrar los valores del display independientemente.

```
CAMBIAR_LED:
    SBIC PINB, PB4    //Si esta encendido la ira a apagar, si esta apgado saltara para encenderlos
    RJMP APAGAR_LED
    SBI PORTB, PB4   //Enciende los leds
    RJMP FIN_CAMBIO //Salir
APAGAR_LED:
    CBI PORTB, PB4
    CALL DELAY
FIN_CAMBIO:
    RET
```

Cambial el estado de la LED, si este encendido salta a la función de apagado