```
.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 está encendido salta a la función de apagado, donde apagara el LED PB4, luego un delay para que le de tiempo de apagar. En caso de que ya este apagado, saltara el apagado y encenderá el led, para luego retornar el reloj.

```
FECHA:
    CBI PORTC, PC4//Se enciende el otro led y se apaga el de el reloj
    SBI PORTC, PC5
    SBRC COUNTER, 0//Mismo proceso solo cambia el modo en las funciones SHOW que ahora mostraran fechas
    CALL SHOW_USEC
    SBRS COUNTER, 0
    CALL SHOW_DSEC
    SBRC COUNTER, 1
    CALL SHOW_UHR
    SBRS COUNTER, 1
    CALL SHOW_DHR
    SBRS COUNTER2, 1
    CALL CAMBIAR_LED
    RJMP SALIDA
```

Este código realiza lo mismo que lo anterior solo que en este caso se enciende el LED en PC5 y se apaga el PC4. Luego realiza el mismo multiplexeado y cambio de LED de medio segundo.

```
CONFIG_RELOJ:
    CBI PORTB, 0
    CBI PORTB, 1
    CBI PORTB, 2
    CBI PORTB, 3
    SBI PORTC, PC4//Se vuelve a encender el LED de reloj y se apaga el de fecha
    CBI PORTC, PC5
    CPI ACTION, 0x00//Ve en si se ha presionado un boton, si se presiona PC0, incrementa y si se presiona PC1 decrementa
    BREQ EXIT_CRELOJ
    CPI ACTION, 0x01
    BREQ INC_CRELOJ
    CPI ACTION, 0x02
    BREQ GO_DEC_CRELOJ
    RJMP SALIDA

GO_DEC_CRELOJ:
    RJMP DEC_CRELOJ
```

Cuando entra a modo configuración de reloj, desactiva todos los transistores y enciende nuevamente PC4 y apaga PC5 para representar que estamos en el reloj. Entonces se verificará el registro ACTION para ver en que estado se encuentra, 0 es que no hubo cambios, 1 que se presionó el botón para incrementar el valor del

registro actual y en 2 se presionó el botón para decrementar. Dependiendo el valor actual ira a una de las tres funciones.

```
EXIT_CRELOJ:
    CPI ACTIONLR, 0x01//Cambia de valor dependiendo si se presiono PC3, y alterna entre izquierda y derecha
    BREQ SHOW_LEFT
    CPI ACTIONLR, 0x02
    BREQ SHOW_RIGHT
    RJMP SALIDA

SHOW_LEFT:
    CALL SHOW_DHR//Muestra los valores solo del lado izquierdo ciclandolos con un delay
    CALL DELAY
    CALL SHOW_UHR
    CALL DELAY
    CALL SHOW_DHR
    RJMP SALIDA

SHOW_RIGHT:
    CALL SHOW_USEC//Muestra los valores solo del lado derecho ciclandolos con un delay
    CALL DELAY
    CALL SHOW_DSEC
    CALL DELAY
    CALL SHOW_USEC
    RJMP SALIDA
```

Si el valor de ACTION es 0, entonces ira al EXIT_CRELOJ. Aquí se ve otra variable llamada ACTIONLR, que ve el estado actual de los display, muestra o el lado izquierdo si su valor es 1 y muestra el derecho en caso de ser 2. Si es 1 entonces mostrara el lado izquierdo que es mostrar las unidades y decenas de horas, de ser 2 mostraría el lado derecho que es las unidades y decenas de minutos.

```
INC_CRELOJ:
    CPI ACTIONLR, 0x01//Dependiendo de los displays que se estan mostrando incrementara el valor de ese display
    BREQ INC_LEFT
    CPI ACTIONLR, 0x02
    BREQ INC_RIGHT
    RJMP SALIDA

INC_LEFT:
    LDI ACTION, 0x00//Reiniciando el cambio del boton
    LDS R16, UHR//Cargando valor de la RAM
    INC R16
    INC COUNTERHR
    CPSE COUNTERHR, MAXDHR//Comparandolo con el maximo para overflow de horas
    RJMP PC+2
    RJMP PC+11
    CPI R16, MAX_UHR//Comparandolo con el maximo para que reinicie en 0
    BRNE GUARDAR_UHR//Si no ha llegado al maximo ira a gurdar el valor en otra funcion
    CLR R16
    STS UHR, R16//Guardando el valor en la RAM
    //Valores del cuarto display
    LDS R16, DHR//Mismo proceso solo cambia el maximo de la hora
    INC R16
    CPI R16, MAX_DHR
    BRNE GUARDAR_DHR
    CLR R16//Reiniciando todos los valor por overflow
    CLR COUNTERHR
    STS UHR, R16
    STS DHR, R16
    RJMP SALIDA

GUARDAR_UHR:
    STS UHR, R16
    RJMP SALIDA

GUARDAR_DHR:
    STS DHR, R16
    RJMP SALIDA
```

En caso de que ACTION sea 1 ira para incrementar. Aquí nuevamente se verifica en que lado se encuentra uno si izquierdo o derecho y luego lo envía a modificar el lado correspondiente. En el caso izquierdo serían las horas y ACTION se convierte en 0 para que no se quede atrapado realizando sumas infinitas al solo haberse dado un botonazo. Se saca el valor de la RAM al R16 y se incrementa en 1, al mismo tiempo que se incrementa COUNTERHR, que es cuantas horas han pasado en total. Luego comparamos si las horas totales ya llegaron a 24 para reiniciar todo a 0 por overflow o seguimos de largo. Si no ha llegado al limite entonces pasa la siguiente línea que luego se salta el salto de 11 líneas. Ahora verificamos si la hora ya llego a 10 para reiniciarla en 0 y sumar 1 a las decenas, en caso de no haber llegado al límite entonces ira a una función donde se guardará el valor y saldrá al inicio del main_loop. Cuando se incrementa decenas realiza la misma acción. En caso de que si haya sido 24 la igualación de COUNTERHR entones se saltara el +2 e ira a saltar 11 líneas, donde aquí se reiniciaran todos los valores a 0 por overflow.

```
INC_RIGHT:
    LDI ACTION, 0x00//Mismo proceso ahora que del lado de los segundos, solo mas simplificado
    LDS R16, USEC   //ya que los limites son numeros exactos de 10 no hay necesidad de un contador extra
    INC R16
    CPI R16, MAX_USEC
    BRNE GUARDAR_USEC
    CLR R16
    //Valores en segundo display
    STS USEC, R16
    LDS R16, DSEC
    INC R16
    CPI R16, MAX_DSEC
    BRNE GUARDAR_DSEC
    CLR R16
    STS DSEC, R16
    RJMP SALIDA

GUARDAR_USEC:
    STS USEC, R16
    RJMP SALIDA

GUARDAR_DSEC:
    STS DSEC, R16
    RJMP SALIDA
```

Ahora si el incremento era del lado derecho entonces se modifican los minutos. Nuevamente ACTION se vuelve 0, para evitar bucles y se suma uno a las unidades. En este caso al ser 60 un numero exacto de 10 no es necesario colocar un contador y simplemente limpiaremos los registros cuando las unidades lleguen a diez o cuando la decena llegue a 6 significando 60 minutos. Siempre que no se cumplan los limites R16 ira las funciones para guardar los valores nuevamente en los espacios de la RAM.

```
DEC_CRELOJ:
    CPI ACTIONLR, 0x01//Cambiara a quien restara dependiendo de cual se este mostrando
    BREQ DEC_LEFT
    CPI ACTIONLR, 0x02
    BREQ DEC_RIGHT
    RJMP SALIDA

DEC_LEFT:
    LDI ACTION, 0x00//Mismo codigo solo cambian los INC a DEC y los limites
    LDS R16, UHR
    DEC R16
    DEC COUNTERHR
    CPSE COUNTERHR, MINUNDER//El limite seria cuando llegue el valor a 0
    RJMP PC+2
    RJMP PC+11
    CPI R16, MIN_UNDER
    BRNE GUARDAR_UHR
    LDI R16, 9// Ahora inicia en 9 por el underflow
    STS UHR, R16
    //Valores del cuarto display
    LDS R16, DHR
    DEC R16
    CPI R16, MIN_UNDER//Cuando llegue a cero reiniciaria los valores a 23 hrs
    BRNE GUARDAR_DHR
    LDI R16, 0x18
    MOV COUNTERHR, R16
    DEC COUNTERHR
    LDI R16, 3
    STS UHR, R16
    LDI R16, 2
    STS DHR, R16
    RJMP SALIDA
```

Si ACTION es 2 entones entramos al modo para decrementar. Nuevamente verificamos si nos encontramos en el lado izquierdo o derecho para luego ver los limites al decrementar. En el caso de ser el lado izquierdo estamos eh horas. Ahora en vez de incrementar los registros luego de sacarlos de la RAM los decrementaremos. Y el límite ahora no es 24 sino -1 ya que para hacer underflow es necesario pasar por 0. Y lo que también cambia es que si llega a los valores mínimos en vez de volver los display 0 el de unidades comienza en 9 en su propio ciclo, pero si los dos son cero entonces se comenzara en el valor 23, dos en la decena y tres en las unidades, modificando también COUNTERHR, para que lleve el registro.

```
DEC_RIGHT:
    LDI ACTION, 0x00
    LDS R16, USEC//Mimso codigo solo cambiando los limites y el valor inicial
    DEC R16
    CPI R16, MIN_UNDER
    BRNE GUARDAR_USEC
    LDI R16, 9
    //Valores en segundo display
    STS USEC, R16
    LDS R16, DSEC
    DEC R16
    CPI R16, MIN_UNDER
    BRNE GUARDAR_DSEC
    LDI R16, 5
    STS DSEC, R16
    RJMP SALIDA
```

Es lo mismo que INC_RIGHT solo que INC se vuelve DEC y los limites se convierten en -1. Los valores iniciales son diferentes de igual forma, las unidades individuales regresan a ser 9 y las decenas a 5.

```
CONFIG_FECHA:
    CBI PORTC, PC4//Se enciende LED de fecha y se apaga LED de reloj
    SBI PORTC, PC5
    CPI ACTION, 0x00
    BREQ GO_EXIT_CRELOJ//Mismo proceso que el codigo de cambio de reloj, solo se modifica lo que muestran las funciones SHOW
    CPI ACTION, 0x01//  y los limites para UNDERFLOW y OVERFLOW
    BREQ INC_CFECHA
    CPI ACTION, 0x02
    BREQ GO_DEC_CFECHA
    RJMP SALIDA

GO_DEC_CFECHA:
    RJMP DEC_CFECHA

GO_EXIT_CRELOJ:
    RJMP EXIT_CRELOJ
```

Ahora si el MODO era 3 se ira a configurar la fecha pero el código principal realiza lo mismo que en configuración de reloj lo que cambian son los incrementos y decrementos de los valores por los botones.

```
INC_CFECHA:
    CPI ACTIONLR, 0x02
    BREQ INC_LEFT_F
    CPI ACTIONLR, 0x01
    BREQ INC_RIGHT_F
    RJMP SALIDA

INC_LEFT_F:
    LDI ACTION, 0x00
    LDS R17, PMES// Se carga la tabla para limite de dias para poder realizar el overflow
    LDI ZL, LOW(LIMTMES << 1)
    LDI ZH, HIGH(LIMTMES << 1)
    ADD ZL, R17
    ADC ZH, R1
    LPM R17, Z
    //Comienza Fecha
    LDS R16, UDIA
    INC R16
    INC COUNTERDIA
    CPSE COUNTERDIA, R17//Se compara si llego al maximo de dias del mes actual
    RJMP PC+2
    RJMP PC+11
    CPI R16, MAX_USEC
    BRNE GUARDAR_UDIA
    CLR R16
    STS UDIA, R16
    //Valores en segundo display
    LDS R16, DDIA
    INC R16
    CPI R16, MAX_DSEC
    BRNE GUARDAR_DDIA
    LDI R16, 0x01//Siempre comienza en 1 el overflow porque no hay dias 0
    CLR COUNTERDIA
    STS UDIA, R16
    CLR R16
    STS DDIA, R16
    RJMP SALIDA
```

Ahora para incrementar los valores del lado izquierdo tenemos a los días. La diferencia en este caso es la carga de la tabla en Z de los límites de días por cada mes. Ahora en vez de tener un mismo registro en comparación con el contador ahora tenemos un registro que varía dependiendo del mes. COUNTERDIA ahora se iguala a R17 que es donde se carga el valor del día según el mes en el que nos encontremos, febrero 28, diciembre 31 y etc. Luego de eso sigue lo mismo solo que ahora las decenas al dar la vuelta por overflow comienzan en y las unidades en 1 ya que no hay días 0.

```
GUARDAR_UDIA:
    STS UDIA, R16
    RJMP SALIDA

GUARDAR_DDIA:
    STS DDIA, R16
    RJMP SALIDA

INC_RIGHT_F:
    LDI ACTION, 0x00
    LDS R16, UMES//Mismo codigo solo que en este caso el limite es 12 para los meses
    INC R16
    LDS R17, PMES
    INC R17
    STS PMES, R17
    CPSE R17, COUNTERMES
    RJMP PC+2
    RJMP PC+11
    CPI R16, MAX_UHR
    BRNE GUARDAR_UMES
    CLR R16
    STS UMES, R16
    LDS R16, DMES
    INC R16
    CPI R16, 0x02
    BRNE GUARDAR_DMES
    LDI R16, 0x01//Tambien empieza desde 1
    STS UMES, R16
    CLR R16
    STS PMES, R16
    STS DMES, R16
    RJMP SALIDA

GUARDAR_UMES:
    STS UMES, R16
    RJMP SALIDA

GUARDAR_DMES:
    STS DMES, R16
    RJMP SALIDA
```

En el caso de ser incremento derecho tenemos a los meses. En este caso además de los ya repetido múltiples que son los mismos cambios tenemos que los meses solo llegan a 12, a diferencia del 60 de minutos. Por lo que en este caso también hay que colocarle un counter(COUNTERMES), que llevará el registro para la tabla y para reiniciar los valores en 01.

```
DEC_CFECHA:
    CPI ACTIONLR, 0x02//Decremente el valor del display
    BREQ DEC_LEFT_F
    CPI ACTIONLR, 0x01
    BREQ DEC_RIGHT_F
    RJMP SALIDA

DEC_LEFT_F:
    LDI ACTION, 0x00
    LDS R17, PMES//Solo cambias los limites para cuando iniciaran los dias en unas funciones extras, luego es lo mismo que en reloj
    LDI ZL, LOW(LIMTMES << 1)
    LDI ZH, HIGH(LIMTMES << 1)
    ADD ZL, R17
    ADC ZH, R1
    LPM R17, Z
    //Comienza Fecha
    LDS R16, UDIA
    DEC R16
    DEC COUNTERDIA
    CPSE COUNTERDIA, MINUNDER
    RJMP PC+2
    RJMP PC+11
    CPI R16, MIN_UNDER
    BRNE GO_GUARDAR_UDIA
    LDI R16, 9
    STS UDIA, R16
    //Valores en segundo display
    LDS R16, DDIA
    DEC R16
    CPI R16, MIN_UNDER
    BRNE GO_GUARDAR_DDIA
    MOV COUNTERDIA, R17//Compara con la tabla el dia actual
    DEC COUNTERDIA//Decremento o no podra hacer overflow luego
    CPI R17, 28
    BREQ FEBRERO
    CPI R17, 30
    BREQ MES_PAR
    CPI R17, 31
    BREQ MES_IMPAR
    RJMP SALIDA
```

Lo mismo que INC_LEFT, solo cambian los limites y los valores iniciales que en este caso si son importantes. Ya que a diferencia del underflow en reloj los valores máximos de los días cambian dependiendo del mes por lo que tenemos que igualar par ver en que mes se encuentra y así poder colocar el nuevo valor luego de llegar a 0.

```
FEBRERO://Si esta en mes 2, el underflow te devolvera al dia 28
    LDI R16, 8
    STS UDIA, R16
    LDI R16, 2
    STS DDIA, R16
    RJMP SALIDA

MES_PAR:// Si el mes es par, entonces el valor que regresara sera 30
    LDI R16, 0
    STS UDIA, R16
    LDI R16, 3
    STS DDIA, R16
    RJMP SALIDA

MES_IMPAR:// Si el mes es impar, entonces el valor que regresara sera 31
    LDI R16, 1
    STS UDIA, R16
    LDI R16, 3
    STS DDIA, R16
    RJMP SALIDA
```

En este caso hay tres situaciones. Si los días son de febrero entonces se cambia el valor a 28, si los días terminan en mes con días par (exceptuando febrero), entonces los valores serán 30 y si el mes termina con días impar entonces será 31.

```
GO_GUARDAR_UDIA:
    RJMP GUARDAR_UDIA

GO_GUARDAR_DDIA:
    RJMP GUARDAR_DDIA

DEC_RIGHT_F:// Solo cambia el hecho que INC es ahora DEC y los limites para realizar underflow son -1
    LDI ACTION, 0x00
    LDS R16, UMES
    DEC R16
    LDS R17, PMES
    DEC R17
    STS PMES, R17
    CPSE R17, MINUNDER
    RJMP PC+2
    RJMP PC+11
    CPI R16, MIN_UNDER
    BRNE GO_GUARDAR_UMES
    LDI R16, 9
    STS UMES, R16
    LDS R16, DMES
    DEC R16
    CPI R16, -1
    BRNE GO_GUARDAR_DMES
    LDI R16, 0x02// El valor luego de hacer underflow seria 12
    STS UMES, R16
    LDI R16, 1
    STS DMES, R16
    LDI R16, 11
    STS PMES, R16
    RJMP SALIDA

GO_GUARDAR_UMES:
    RJMP GUARDAR_UMES

GO_GUARDAR_DMES:
    RJMP GUARDAR_DMES
```

Para el lado derecho si llegan a ser las mismas modificaciones de siempre de los limites a la hora de restar a los registros, cambiar INC a DEC y los valores iniciales de luego de cumplir los límites.