# List of actions

On the boot the device will load the default configuration that is programmed into the device.

In this default, the following instructions are made:

* Including the m32def.inc documentation
* Defining variables in the registers
* Declaring the origins for the interrupts and resets.

Afterwards when this is completed the program will run.

First off the program will go to the **init** label for initiating the Crystal timer and the 00Clock

In this initiation the program will:

* Set the interval of the Crystal to half a second.
* Set the **alarmSet** to 0, this is alarm off
* Set the **hour** counter to 00
* Set the **minute** counter to 00
* Set the **second** counter to 00
* Set the **alarm hour** counter to 00
* Set the **alarmMinute** counter to 00
* Set **editLevel** input to 0

Afterwards the is ready to go and will go to the **loop** label to wait for users input.

When the 00Clock loops for the first time. The **editLevel** value is still on zero. While this is set on zero, the program will not continue counting the time. It will stay blinking on ’00:00:00alarm’ for hours.

In this loop, the SW1 button would be looked into. If the value of SW1 stays 0 for more than 3 seconds. Then the next step is activated.

On the next step the **editLevel** value would be incremented with one. At this stage the hours can be set, and the hours indicator would blink.

By Holding the SW0 button the hours would increment every one second.

By pushing the SW0 button for 0,25 seconds the hour would increment with one.

When the correct time is set, the user needs to Hold SW1 again for 0,5 second. At this point the process would repeat itself until the seconds are set and the **editLevel** value is 3.

When the **editLevel** value is 3 (or higher) the Alarm can be set. The same steps would be made only the alarm indicator would be blinking. This continues until the **editLevel** value hits 7. At this point the Alarm hours and minutes are set. And the program can continue to the set alarm mode.

The clock is running in the background.

At this stage you can either chose to activate the alarm or disable it. To Toggle the alarm you push the SW0 button, and to Disable it you will need to push the SW1 button.

The **editLevel** would be set to the max value and the clock would now function as it should be.

For every 60 second, the minute would increment and the seconds reset to 0.

For every 60 minutes, the hour would increment, and the minutes reset to 0.

For Every 24 hours, the hours would be reset to 0.

When the user is holding the SW0 button for 0,5 seconds or longer, the alarm time would be shown.

To enable or disable the alarm, the user needs to hold the SW1 button for 0,5 seconds.

(Extra)

To adjust the Alarm clock, the user needs to hold SW1 for 3 seconds. At this point the **editLevel** value is brought back to value 4. At this state the user can edit the alarm clock.

**EditLevel** info table:

|  |  |  |  |
| --- | --- | --- | --- |
| Value | Modes | What it would do | view |
| 0 | Startup mode: | The hours, minutes, and seconds would blink every second second. | Clock mode |
| 1 | Setup hours | The hours indicator would blink every second until the hours are set. |
| 2 | Setup minutes | The minutes indicator would blink every second until the minutes are set. |
| 3 | Setup seconds | The seconds indicator would blink every second until the seconds are set. This increments every 10 seconds due to catching up real time seconds. From this point, the clock would be running. |
| 4 | Setup Alarm: | The hours, minutes, seconds and alarm-indicator would blink every second. | Alarm Mode |
| 5 | Setup Alarm-Hour | The hours indicator would blink every second until the hours are set. The alarm-indicator is shown |
| 6 | Setup Alarm-Minutes | The minutes indicator would blink every second until the minutes are set. The alarm-indicator is shown |
| 7 | Enable/disable Alarm | The user can now enable or disable the alarm |
| 8 | Setup-completed | The user can see the alarm time and enable/disable the alarm. (extra: When holding the enable alarm button SW1 for more than 3 seconds the user-adjust value goes to value 4.) | Clock mode  / Alarm mode when asked for) |

# Variables & SUbroutines

## variables

|  |  |  |
| --- | --- | --- |
| Name | Register | Purpose |
| **editLevel** |  | The variable that determines the step you are in, in the progress of setting up the 00Clock. The value is between 0 and 8. |
| **hour** |  | The hour variable determines the hour. From the radius of 0 to 23. |
| **minute** |  | The minute variable determines the minutes. From the radius of 0 to 59. |
| **seconds** |  | The seconds variable determines the seconds. From the radius of 0 to 59. |
| **alarmHour** |  | The alarmHour variable determines the hour that the alarm would be set on. From the radius of 0 to 23. |
| **alarmMinute** |  | The minute variable determines the minutes that the alarm would be set on. From the radius of 0 to 59. |
| **alarmSet** |  | The variable that determines if the alarm is set on or off. The values are either 0x00 or 0xFF. |
| **SW0counter** |  | The variable that counts the amount of loops that SW0 is pushed into. |
| **SW1counter** |  | The variable that counts the amount of loops that SW1 is pushed into. |

## subroutines

|  |  |  |
| --- | --- | --- |
| Name | As known as | What it will do |
| **init** | Initiate | * Set the **alarmSet** to 0, this is alarm off * Set the **hour** counter to 00 * Set the **minute** counter to 00 * Set the **second** counter to 00 * Set the **alarmHour** counter to 00 * Set the **alarmMinute** counter to 00 * Set **editLevel** input to 0 |
| **incHour** | Increment hour | * Increment the hour. * If the hour is greater or equal to 24, set hour to 00. * Return to last position |
| **incMinute** | Increment minute | * Increment the minute. * If the minutes are greater or equal to 60, set minutes to 00 and call the incHour label * Return to last position |
| **IncSecond** | Increment seconds | * Increment the seconds. * If the seconds are greater or equal to 60, set the seconds to 00 and call the incMinute label. * Return to last position |
| **incAlarmHour** | Increment hours on alarm mode | * Increment the alarmHour. * If the alarmHour is greater or equal to 24, set alarmHour to 00. * Return to last position |
| **incAlarmMinute** | Increment minutes on alarm mode | * Increment the alarmMinute. * If the alarmMinute is greater or equal to 60, set alarmMinute to 00. * Return to last position |
| **incEditLevel** | Increments editing level | * Increment editing level * If editing level is equal or greater than 9, set editing level to 4 * Return to last position |
| **checkEditLevel** | Check the edit level | * Checks the editLevel * If editLevel is equal to 0, go to startup label * If editLevel is equal to 1, go to setHour label * If editLevel is equal to 2, go to setMinute label * If editLevel is equal to 3, go to setSecond label * If editLevel is equal to 4, go to setAlarmStartup label * If editLevel is equal to 5, go to setAlarmHour label * If editLevel is equal to 6, go to setAlarmMinute label * If editLevel is equal to 7, go to setAlarmMode label * If editLevel is equal to 8, go to checkSetAlarm label |
| **checkSetAlarm** | Check if the alarm is set | * Checks if alarm is set. * If alarm is set check hour, minutes, to the actual time. When it equal, go to displayAlarmSound |
| **startup** | Startup mode | * Blinking the hours, minutes and seconds every half second * When the SW1counter is greater than 6, go to incEditLevel label |
| **setHour** | Setting hour | * Blinking the hour every half second. * When SW0counter is greater than 1, go to incHour label. * When the SW1counter is greater than 6, go to incEditLevel label |
| **setMinute** |  | * Blinking the minute every half second. * When SW0counter is greater than 1, go to incMinute label. * When the SW1counter is greater than 6, go to incEditLevel label |
| **setSecond** |  | * Blinking the second every half second. * When SW0counter is greater than 1, go to incSecond label. Repeat 10 times. * When the SW1counter is greater than 6, go to incEditLevel label |
| **setAlarmStartup** |  | * Blinking the hours, minutes, seconds and alarm every half second * When the SW1counter is greater than 6, go to incEditLevel label |
| **setAlarmHour** |  |  |