Alarm

# Introduction

Alarm provides the mechanism to schedule activities for the future which occur periodically or are executed at once. This application note demonstrates the alarm functionality in Talaria TWO.

# Relevant APIs

## alarm\_init()

Initializes the alarm.

|  |
| --- |
| alarm\_init(); |

## alarm\_set\_time ()

Sets the system time.

|  |
| --- |
| alarm\_set\_time (); |

## sntp\_setservername ()

Initializes one of the NTP servers via the IP address of the Talaria TWO module.

|  |
| --- |
| sntp\_setservername (); |

## sntp\_init()

Initializes the SNTP of the Talaria TWO module. It sends out a request instantly or after sntp\_startup\_delay(func).

|  |
| --- |
| sntp\_init (); |

## sntp\_stop ()

Stops the Talaria TWO module.

|  |
| --- |
| sntp\_stop (); |

## uart\_puts ()

Writes a string to the serial port.

|  |
| --- |
| uart\_puts (); |

## uart\_flush()

Flushes the output buffer.

|  |
| --- |
| uart\_flush(); |

## uart\_close()

Closes the serial port.

|  |
| --- |
| uart\_close(); |

## uart\_getchar()

Reads one character from the serial port.

|  |
| --- |
| uart\_getchar (); |

## uart\_putchar()

Writes one character to the serial port.

|  |
| --- |
| uart\_putchar(); |

## uart\_set\_event\_callback()

Sets event callback.

|  |
| --- |
| uart\_set\_event\_callback(); |

# Code Walkthrough

## Connecting to a Wi-Fi network

To connect to a Wi-Fi network, wcm\_create()API from the Wi-Fi Connection Manager are used. Initially, the Wi-Fi network interface is created using wcm\_create().

|  |
| --- |
| h = wcm\_create(NULL); |

wcm\_connect\_to\_network()reads the configurations and connects to network.

|  |
| --- |
| rval = wifi\_connect\_to\_network(&h, WCM\_CONN\_WAIT\_INFINITE, &wcm\_connect\_success);  if(rval < 0) {  os\_printf("\nError: Unable to connect to network\n");  return 0;  } |

## Validating Date

validdate\_date() validates the date in month, year, and date, and returns informing whether the parameters which are set are valid or not.

|  |
| --- |
| void sntp\_setservername(u8\_t idx, char \*server);  uint8\_t validdate\_date(uint32\_t yy,uint32\_t mm,uint32\_t dd)  {  //check year  if(yy>=1900 && yy<=9999)  {  //check month  if(mm>=1 && mm<=12)  {  //check days  if((dd>=1 && dd<=31) && (mm==1 || mm==3 || mm==5 || mm==7 || mm==8 || mm==10 || mm==12))  os\_printf("Date is valid.\n");  else if((dd>=1 && dd<=30) && (mm==4 || mm==6 || mm==9 || mm==11))  os\_printf("Date is valid.\n");  else if((dd>=1 && dd<=28) && (mm==2))  os\_printf("Date is valid.\n");  else if(dd==29 && mm==2 && (yy%400==0 ||(yy%4==0 && yy%100!=0)))  os\_printf("Date is valid.\n");  else  {  os\_printf("Day is invalid.\n");  return 1;  }  }  else  {  printf("Month is not valid.\n");  return 1;  }  }  else  {  return 1;  }  return 0;  } |

## Validating Time

validdate\_time() validates the time in hour, minute, and seconds.

|  |
| --- |
| uint8\_t validdate\_time(int32\_t hh, int32\_t mi, int32\_t se)  {  if((hh < 0 || hh >23 ) || (mi < 0 || mi > 59) ||(se < 0 || se > 59)) {  os\_printf("Time is Not valid.\n");  return 1;  }  else {  os\_printf("Time is valid.\n");  return 0;  }  } |

## Get NTP time

ntp\_time\_get()is used to get the NTP time. Here, sntp\_setservername() initializes one of the NTP servers via the IP address of the Talaria TWO module and then sntp\_init() initializes the SNTP of the Talaria TWO module.

|  |
| --- |
| unsigned int ntp\_time\_get()  {  int times = 0;  unsigned int time\_now;  if(NULL != ntp\_srv\_name)  sntp\_setservername(0, (char \*)ntp\_srv\_name);  sntp\_init();  do {  os\_printf("waiting for sntp, times=%d\n", times++);  time\_now = sntp\_time();  if(time\_now == 0 && times < 16) {  os\_sleep\_us(2000000, OS\_TIMEOUT\_NO\_WAKEUP);  continue;  }  else  break;  } while(time\_now == 0 && times < 10);  sntp\_stop();  return time\_now;  } |

## Read Input String from UART

uart\_getchar() and uart\_putchar()reads and writes one character from serial port.

|  |
| --- |
| void get\_string(uint8\_t \*buffer, uint32\_t len)  {  int ch;  int chindex = 0;  uint8\_t \*ptr = buffer;  while(chindex < len) {  ch = uart\_getchar(uarthandle);  if(ch == '\r' || ch == '\n') {  ptr[chindex++] = '\0';  return;  }  else if(ch == '\b')  {  uart\_putchar(uarthandle, ch);  uart\_putchar(uarthandle, ' ');  uart\_putchar(uarthandle, ch);  chindex--;  }  else {  uart\_putchar(uarthandle, ch);  ptr[chindex++] = ch;  }  }  } |

## Alarm Callback

User defined function alarm\_user\_cb()is used to set the alarm ID and the argument. It prints the alarm ID and name, once the alarm is set.

|  |
| --- |
| void  alarm\_user\_cb(uint32\_t id , uint8\_t \*arg)  {  char buffer[128];  os\_printf("alarm\_user\_cb");  snprintf(buffer, 256, "\r\n \*\*\*\*\*Alarm:Id-%d:Name=%s \*\*\*\*\*", id, arg);  uart\_puts(uarthandle, buffer); |

## Alarm Display Callback

User defined function alarm\_display\_cd() is used to display the alarm information by printing the following:

1. Alarm ID
2. Name
3. Time
4. Type
5. If it is a repeating alarm or not.

|  |
| --- |
| void  alarm\_display\_cb(struct alarm\_info \*ainfo)  {  os\_printf("alarm\_display\_cb");  snprintf((char \*)buffer, 256," \r\nAlarm ID:%d \r\nnName:%s\r\nAlarm Time:%s\r\nAlarm Type:%s\r\nRepeat:%s\r\n",  ainfo->alarm\_id, ainfo->alarm\_name, time\_tostr((time\_t \*)&ainfo->alarm\_timesec),  (ainfo->alarm\_type == ALARM\_TYPE\_DAILY) ?"DAILY":"WEEKLY",  (ainfo->alarm\_repeat == 1) ? "Yes":"No");  uart\_puts(uarthandle, buffer);  } |

## Suspend the System

uart\_suspend\_enable()enables suspend and waits for the system to wakeup, where os\_sleep\_us() suspends execution for the specified number of microseconds and uart\_flush() flushes the output buffer.

|  |
| --- |
| static void  uart\_suspend\_wait(struct uart \*u)  {  suspend=1;  uart\_suspend\_enable(uarthandle);  while(suspend)  {  os\_sem\_wait(&suspend\_lock);  if(suspend)  uart\_suspend\_enable(uarthandle);  };  os\_sleep\_us(100000, OS\_TIMEOUT\_NO\_WAKEUP);  uart\_flush(uarthandle);  uart\_close(uarthandle);  uarthandle = uart\_open(115200);  uart\_puts(uarthandle, "Out of Sleep !!\r\n");  } |

# Cases Supporting the Alarm Menu

## Case 0: Sleep Mode

Puts Talaria TWO in sleep mode.

|  |
| --- |
| case 0: /\*Sleep\*/  {  uart\_flush(uarthandle);  uart\_set\_event\_callback(uarthandle, handle\_event, NULL);  uart\_suspend\_wait(uarthandle);  break;  } |

## Case 1: Wi-Fi Connection Status

Checks for Wi-Fi connection status. If the link is up, then by using ntp\_time\_get() NTP time in seconds will be printed. However, if the Wi-Fi link is down, ntp\_time\_get() fails.

|  |
| --- |
| case 1: /\*Get NTP time\*/  {  /\*check wifi connection status\*/  wcm\_get\_status(alarm\_wcm\_handle, &wcmstat);  if(wcmstat.link\_up) {  tim\_now = ntp\_time\_get();  os\_printf("\r\n Ntp time:%d Sec", tim\_now);  snprintf(buffer, 256, "\r\n NTP Time:%d sec\r\n", tim\_now);  uart\_puts(uarthandle, buffer);  }  else {  snprintf(buffer, 256, "\r\n WIFI link down: Ntp time get failed");  os\_printf("%s", buffer);  uart\_puts(uarthandle, buffer);  }  break;  } |

## Case 2: System Time

gettimeofday()gets system time in seconds.

|  |
| --- |
| case 2: /\*Get System time Seconds\*/  gettimeofday(&now, NULL);  os\_printf("\r\n time:%lld", now.tv\_sec);  snprintf(buffer, 256, "\r\n System Time:%lld sec\r\n", now.tv\_sec);  uart\_puts(uarthandle, buffer);  break; |

## Case 3: System Date and Time

gettimeofday() gets system date and time, prints the date, day, time in hour minute and seconds as per local time.

|  |
| --- |
| case 3: /\*Get System Date and time\*/  {  struct tm \*tm;  gettimeofday(&now, NULL);  os\_printf("\r\n time:%lld", now.tv\_sec);  tm = localtime(&now.tv\_sec);  os\_printf("\r\n timew:%d", tm->tm\_hour);  strftime(buffer, 64, "%a %b %e %T %Y\n", tm);  uart\_puts(uarthandle, buffer);  os\_printf("\r\n Date:%s",time\_tostr(&now.tv\_sec));  break;  } |

## Case 4: Set System Time

Here, wcm\_get\_status()checks the Wi-Fi connection status. If the link is up, ntp\_time\_get() gets the NTP time. If the link is down, ntp\_time\_get()fails to get the NTP time.

Only if the NTP time is captured, the alarm can be set using alarm\_init(), after which alarm\_set\_time() will replace the system time with the NTP time. This helps set the alarm for present date and time.

|  |
| --- |
| case 4: /\*Set Sytem time with NTP time\*/  {  /\*check wifi connection status\*/  wcm\_get\_status(alarm\_wcm\_handle, &wcmstat);  if(wcmstat.link\_up) {  tim\_now = ntp\_time\_get();  /\*initilaise the alarm only after setting the time\*/  if(0 != tim\_now) {  alarm\_init();  alarm\_set\_time((uint64\_t)tim\_now);  }  }  else {  snprintf(buffer, 256, "\r\n WIFI link down: Ntp time get failed");  os\_printf("%s", buffer);  uart\_puts(uarthandle, buffer);  }  break;  } |

## Case 5: Set System Time in Seconds

Sets system time in seconds. alarm\_init() initializes the alarm and alarm time is set using alarm\_set\_time().

|  |
| --- |
| case 5: /\*Set Sytem time in seconds\*/  {  snprintf(buffer, 256, "\r\n Time(seconds):");  uart\_puts(uarthandle, buffer);  tim\_now= get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  /\*Initiliase alarm\*/  alarm\_init();  /\*Set Time\*/  alarm\_set\_time((uint64\_t)tim\_now);  break;  } |

## Case 6: Set the Alarm

Sets the alarm with an option to add the following:

1. alarm\_time.tm\_year – Year
2. alarm\_time.tm\_mon – Month (1-12)
3. alarm\_time.tm\_mday – Day (1-31)
4. alarm\_time.tm\_hour – Hour (0-23)
5. alarm\_time.tm\_min – Minute (0-59)
6. alarm\_time.tm\_sec – Second (0-59)
7. alarm\_type – Can be set for:
   1. DAILY (0) or WEEKLY (1)

If the alarm is set for daily, on the configured time, the alarm will be notified to the user on the serial console with a string containing the alarm ID and name, every day at the same time.

If the alarm is set for weekly, on the configured time, the alarm will be notified to the user on the serial console with a string containing the alarm ID and name, every week at the same time.

* 1. Shot (0) or Repeat (1)

If the alarm should be notified to the user only once on a DAILY or WEEKLY basis, then the “Shot” option should be selected.

If the alarm should be periodically notified to the user on a DAILY or WEEKLY basis, then the “Repeat” option should be selected.

If the user enters a wrong value for the prompted options, an ‘Invalid’ followed by the option is printed on the terminal.

For example: ‘Invalid date, time’ is printed on the terminal.

If configuring of the alarm is not successful, ‘Alarm set failed’ is printed. Else, the alarm ID is printed on the terminal.

|  |
| --- |
| case 6: /\*Set Alarm\*/  { struct alarm\_tm alarm\_time;  uint8\_t alarm\_type;  uint8\_t periodic;  snprintf(buffer, 256, "\r\n Year:");  uart\_puts(uarthandle, buffer);  alarm\_time.tm\_year = get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  snprintf(buffer, 256, "\r\n Month(1-12):");  uart\_puts(uarthandle, buffer);  alarm\_time.tm\_mon = get\_num((uint8\_t \*)buffer,256, &status);  if(status < 0)  break;  snprintf(buffer, 256, "\r\n Day(1-31):");  uart\_puts(uarthandle, buffer);  alarm\_time.tm\_mday = get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  if(validdate\_date(alarm\_time.tm\_year, alarm\_time.tm\_mon, alarm\_time.tm\_mday))  {  snprintf(buffer, 256, "\r\n Invalid Date");  uart\_puts(uarthandle, buffer);  break;  }  snprintf(buffer, 256, "\r\n Hour(0-23):");  uart\_puts(uarthandle, buffer);  alarm\_time.tm\_hour = get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  snprintf(buffer, 256, "\r\n Min(0-59):");  uart\_puts(uarthandle, buffer);  alarm\_time.tm\_min = get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  snprintf(buffer, 256, "\r\n Sec(0-59):");  uart\_puts(uarthandle, buffer);  alarm\_time.tm\_sec = get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  if(validdate\_time((int32\_t)alarm\_time.tm\_hour, (int32\_t)alarm\_time.tm\_min, (int32\_t)alarm\_time.tm\_sec))  {  snprintf(buffer, 256, "\r\n Invalid Time");  uart\_puts(uarthandle, buffer);  break;  }  snprintf(buffer, 256, "\r\n Alarm Type(DAILY(0)/WEEKLY(1):");  uart\_puts(uarthandle, buffer);  alarm\_type = get\_num((uint8\_t \*)buffer, 256, &status);  if((status < 0 ) || (alarm\_type < 0 || alarm\_type > 1))  {  status =-1;  break;  }  snprintf(buffer, 256, "\r\n Repeat(One shot(0)/Repeat(1):");  uart\_puts(uarthandle, buffer);  periodic = get\_num((uint8\_t \*)buffer, 256, &status);  if((status < 0)|| (periodic < 0 || periodic > 1) )  {  status =-1;  break;  }  snprintf(buffer, 256, "\r\n Name:");  uart\_puts(uarthandle, buffer);  get\_string((uint8\_t \*)buffer, 256);  os\_printf("\r\n alarm:%d:%d:%d:%d:%d:%d",alarm\_time.tm\_year, alarm\_time.tm\_mon, alarm\_time.tm\_mday,  alarm\_time.tm\_hour, alarm\_time.tm\_min, alarm\_time.tm\_sec);  alarm\_id = alarm\_set(&alarm\_time, alarm\_type, periodic, (uint32\_t)alarm\_user\_cb,  (uint8\_t \*)buffer);  if(alarm\_id < 0) {  snprintf(buffer, 256, "\r\n Alarm set failed:%d\r\n", alarm\_id);  }  else {  snprintf(buffer, 256, "\r\n Alarm ID:%d\r\n", alarm\_id);  }  uart\_puts(uarthandle, buffer);  break;  } |

## Case 7: Delete the Alarm

Deletes the alarm. On checking the alarm ID, if it is more than 0, alarm is deleted. However, if the alarm ID is less than 0, alarm deletion fails.

|  |
| --- |
| case 7: /\*Delete Alarm\*/  { snprintf(buffer, 256, "\r\n Alarm ID to Delete:");  uart\_puts(uarthandle, buffer);  alarm\_id = get\_num((uint8\_t \*)buffer, 256, &status);  if(status < 0)  break;  if(alarm\_delete(alarm\_id) < 0) {  snprintf(buffer, 256, "\r\n Alarm Delete failed:%d\r\n", alarm\_id);  }  else {  snprintf(buffer, 256, "\r\n Alarm Deleted\r\n");  }  uart\_puts(uarthandle, buffer);  break; } |

## Case 8: Display Configured Alarm

Displays the configured alarm.

|  |
| --- |
| case 8: /\*Display Alarm\*/  { alram\_display((uint32\_t)alarm\_display\_cb);  break; }  default:  break; |

# Running the Application

## Programming Talaria TWO

Program alarm.elf (*sdk\_x.y\apps\alarm\bin*) using the Download tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down
   2. ELF Input: Load the alarm.elf by clicking on Select ELF File.
   3. AP Options: Provide the SSID and Passphrase under AP Options to connect to an Access Point.
   4. Programming: Prog RAM or Prog Flash as per requirement.

For more details on using the Download tool, refer to the document: UG\_Download\_Tool.pdf (path: *sdk\_x.y\pc\_tools\Download\_Tool\doc*).

**Note**: x and y refer to the SDK release version. For example: *sdk\_2.5\doc*.

## Expected Output

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWWWAEBuild $Id: git-34e3eddb8 $  np\_conf\_path=/data/nprofile.json ssid=InnoPhase\_AE passphrase=Inno@1234  $App:git-fc1c5cb4  SDK Ver: sdk\_2.5  Alarm Demo App  addr e0:69:3a:00:2c:3c  Adding network: ssid = InnoPhase\_AE : passphrase = Inno@1234  Connecting to added network : InnoPhase\_AE[0.961,039] CONNECT:98:da:c4:73:b7:76 Channel:11 rssi:-32 dBm  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_LINK\_UP  wcm\_notify\_cb to App Layer - WCM\_NOTIFY\_MSG\_ADDRESS  [1.102,314] MYIP 192.168.0.130  [1.102,479] IPv6 [fe80::e269:3aff:fe00:2c3c]-link  Connected to < InnoPhase\_AE > network |

# Evaluating the Application

For this menu-driven application, UART is used as the interface to input the menu options. Launch any of the serial terminal for example GTK term or minicom with the following configurations:

**Note**: This application is supported on both Windows and Linux.

1. Baud Rate: 115200bps
2. Select the required port.

|  |
| --- |
| /dev/ttyUSB\* |

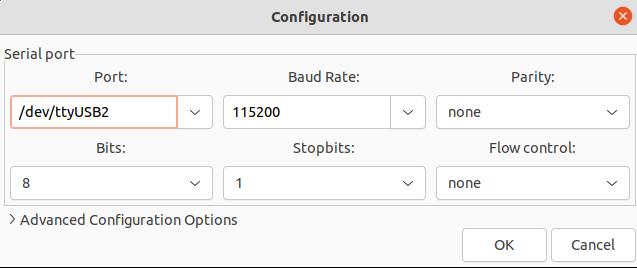


Figure : Configuring the serial terminal

Once the application is flashed, reset the board, and observe the following outputs on the console:

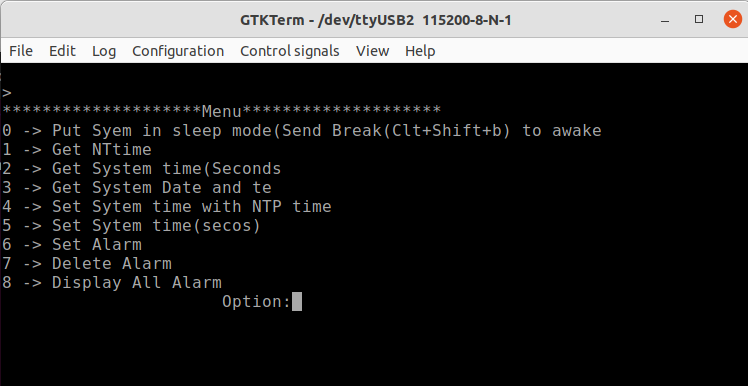


Figure : Menu options - UART terminal

Subsequent sections show the different menu option configurations.

## Put System in Sleep Mode

To put the system in sleep mode, set the option to 0. To wake Talaria TWO up from sleep, send break(Clt+Shift+b).

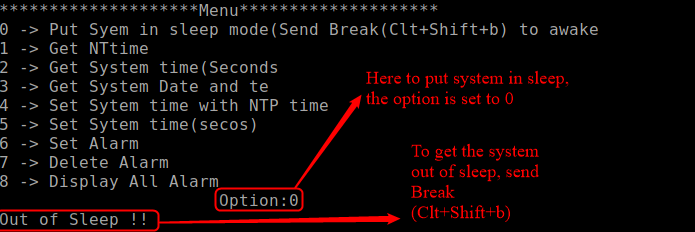


Figure : Put system in sleep mode

## Get NTP Time

To get NTP time in seconds, set the option to 1.

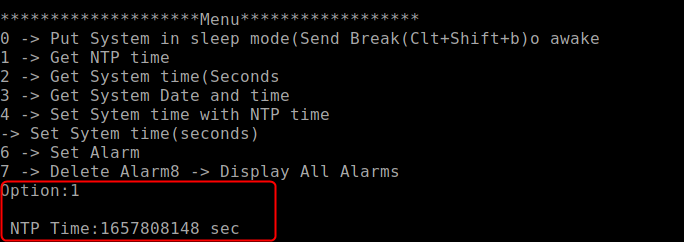


Figure : Get NTP time

## Get System Time

To get system time in seconds, set the option to 2.

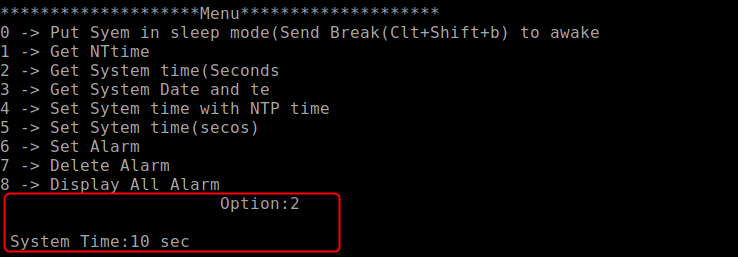


Figure : Get system time

## Get System Date and Time

To get system date and time, set option to 3.

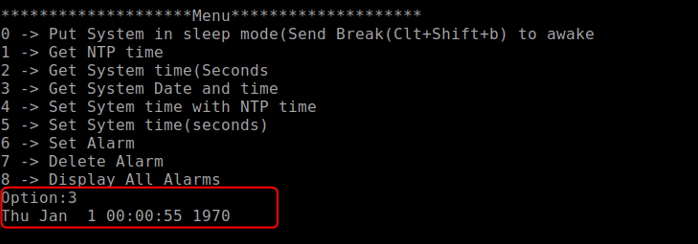


Figure : Get system date and time

## Set System Time with NTP Time

To set system time with NTP time, set the option to 4. Use option 3 to check if the time is set correctly.

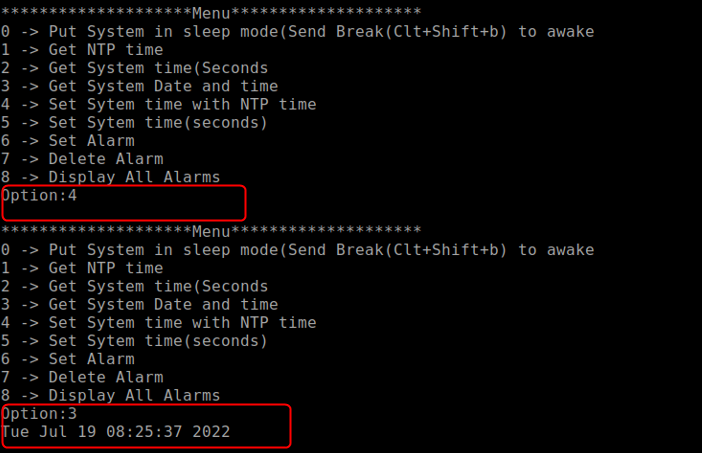


Figure : Set system time with NTP time

## Set System Time (Seconds)

To set system time in seconds, set the option to 5.

**Note**: If the user needs to get back to the default system time, then set the Time(seconds): to 0.

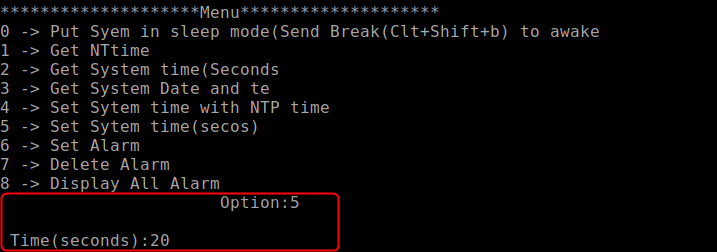


Figure : Set system time in seconds

## Set Alarm

To set the alarm, set the option to 6. Once the alarm is successfully set, the Alarm ID is generated. Multiple alarms can be set using this option.

For more information on setting the alarm, refer section 6.7.

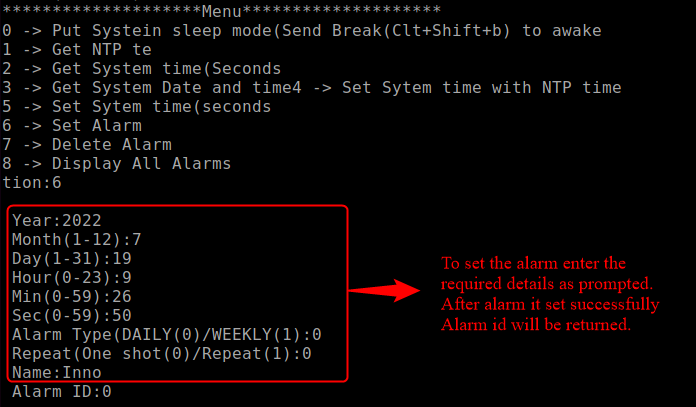


Figure : Set alarm

## Delete Alarm

To delete the alarm, set the option to 7.

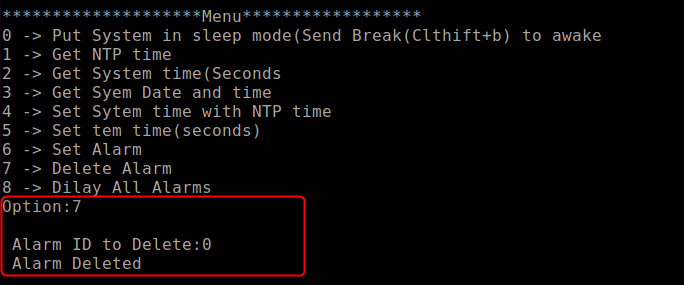


Figure : Delete the alarm

## Display All Alarms

To display all the configured alarms, set the option to 8.

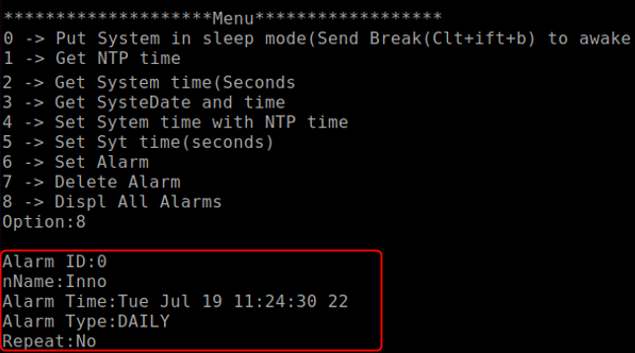


Figure : Display all alarms

## Alarm Expiry

Once the alarm expires , the alarm ID and name is displayed on the terminal, depending upon the configured time added in section 8.7.

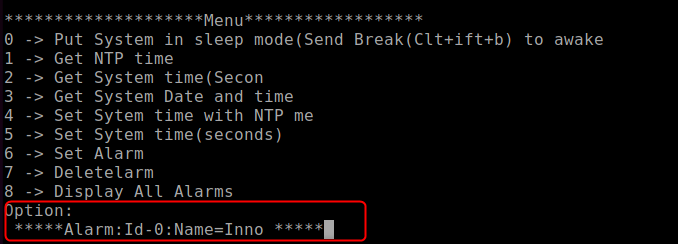


Figure : Alarm expiry