

Assignment 1 Design

Ray Su

B00634512

The design shows the general logic of the interrupt driven monitor in structured English and process flow diagram.

For this assignment, we are asked to design, build and test an interrupt driven monitor which communicate to a VT-100 terminal through UART. User should be able to input command that set or read time and date. Also, user can turn off or set an alarm by giving delta time value.

The program has three layers: application layer, queue layer and UART layer. UART layer is the layer that receive and transmit data through VT-100 terminal. Also, UART layer can receive and handle interrupts. Queue layer is to store data which has received or ready to be transmitted. EnQueue and DeQueue function are the only two function that can be accessed by UART and application. Application layer is to process the received data which is stored in input queue. Also, application layer should send output data by enqueueing to output queue.

The data dictionary for this assignment is included.

Data Dictionary

InQ = * Input queue is a Queue structure that stores input data to be processed, which has element of Head, Tail and array of QueueData *

OutQ = *Output queue is a Queue structure that stores output data to be send out, which has element of Head, Tail and array of QueueData *

UART_STATUS = *A flag that indicate Uart status, has two value: IDLE (0), BUSY (1) *

UART0_DR_R = *UART 0 data register*

Str = * array of char that store the user's input command, has max size 60 *

Clock = * a Systick_Clock data to store time information of application's time *

Alarm = * a Systick_Clock data to store time information of application's alarm *

is_ESC_seq = * the flag that indicates if is receiving in the Esc sequence *

is_alarm_active = * the flag that indicates if alarm is active*

time[] = *the array to store integer of time information from the input string, which has order {hour, minute, second, tens second}*

days_list = *the const 2D array of number of days in each month for leap year or normal year*

time_symbol = *the const array that stores symbol of valid time data*

mon_list = *the const array of the first three letters of each month*

Data Structure

Enums:

- Source: The source of input data
 - UART (0)
 - SYSTICK (1)
- QueueType: The type of queue
 - INPUT (0)
 - OUTPUT (0)

Struct:

- QueueData: Structure of data in queue
 - Source: the source of data
 - Value: the value of data
- Queue: structure of queue to store QueueData
 - Head: the head of queue data, which is an int value indicates head's position in queue
 - Tail: the tail of queue data, which is an int value indicates tail's position in queue
 - Queue: array of QueueData, which has size of 8.
- Systick_Clock: structure of systick clock to store time information
 - t_sec: integer of tenth of second
 - sec: integer of second
 - min: integer of minute
 - hour: integer of hour
 - day: integer of day
 - month: integer of month
 - year: integer of year

Structured English for EnQueue function

SELECT CASE

CASE 1 enqueue to InQ

IF InQ NOT FULL THEN

WRITE data to InQ HEAD

INCREASE InQ HEAD

RETURN TRUE

END IF

END CASE 1

CASE 2 enqueue to OutQ

IF OutQ NOT FULL THEN

TURN OFF UART0 TRANSMIT interrupt

IF UART_SATAUS == BUSY THEN

WRITE data to OutQ HEAD

INCREASE OutQ HEAD

ELSE

SET UART_STATUS to BUSY

WRITE data to UART0_DR_R

END IF

TURN ON UART0 TRANSMIT interrupt

RETURN TRUE

END IF

END CASE 2

END SELECT CASE

RETURN FALSE

Structured English for DeQueue function:

SELECT CASE

 CASE 1 dequeue InQ

 IF InQ NOT EMPTY THEN

 READ data from InQ TAIL

 INCREASE InQ TAIL

 ELSE

 RETURN FALSE

 END IF

 END CASE 1

 CASE 2 dequeue OutQ

 IF OutQ NOT EMPTY THEN

 READ data from OutQ TAIL

 INCREASE OutQ TAIL

 ELSE

 Return FALSE

 END IF

 END CASE 2

END SELECT CASE

RETURN TRUE

Structured English for CheckInputQueue function:

IF there is data can be DEQUEUE from InQ THEN

IF data SOURCE is UART THEN

IF is_ESC_seq EQUAL TRUE THEN

IF data is the END of ESC sequence THEN

SET is_ESC_seq to FALSE

END IF

ENQUEUE data to OutQ

ELSE IF data is COMMON CHARACTER THEN

*common character: from SPACE (32) to '~' (126) *

IF str is NOT FULL THEN

ENQUEUE data to OutQ

IF data is a LOWER-CASE letter

CONVERT data to UPPER-CASE letter

END IF

STORE data to str

END IF

ELSE IF data is a BACKSPACE THEN

IF str is NOT EMPTY THEN

REMOVE last character

ENQUEUE data to OutQ

END IF

ELSE IF data is ENTER THEN

PROCESS str *see Structured English for process str*

ELSE IF data is ESC THEN

SET is_ESC_seq to TRUE

ENQUEUE data to OutQ

END IF

```
ELSE IF data SOURCE is SYSTICK THEN  
    INCREASE clock by 0.1 second  
    CHECK alarm  
END IF
```

Structured English for process str

IF str START with 'TIME'

IF SIZE of str is MINIMUM SIZE (4) *has no parameter*

OUTPUT Clock TIME

ELSE IF SIZE of str <= MAXIMUM SIZE (15) *has parameter*

DECODE str TIME *See DecodeTime process flow diagram*

IF str has NO ERROR

SET decoded value to Clock TIME

OUTPUT Clock TIME

ELSE

OUTPUT ERROR

END IF

END IF

ELSE IF data START with 'DATE'

IF SIZE of str is MINIMUM SIZE (4) *has no parameter*

OUTPUT Clock DATE *See DecodeDate process flow diagram*

ELSE IF SIZE of str EQUAL to MAXIMUM SIZE (15) *has parameter*

DECODE str DATE

IF str has NO ERROR

SET decoded value to Clock DATE

OUTPUT Clock DATE

ELSE

OUTPUT ERROR

END IF

ELSE

OUTPUT ERROR

END IF

ELSE IF data START with 'ALARM'

IF SIZE of str is MINIMUM SIZE (5) *has no parameter*

TURN OFF Alarm

ELSE IF SIZE of str <= MAXIMUM SIZE (16) *has parameter*

DECODE str TIME *See DecodeTime process flow diagram*

IF str has NO ERROR

SET Alarm EQUAL to Clock

ADD decoded value to Alarm TIME

OUTPUT Alarm TIME

ELSE

OUTPUT ERROR

END IF

END IF

END IF

Structured English for check alarm

IF alarm is ACTIVE THEN

 IF Alarm HOUR EQUAL to Clock HOUR THEN

 IF Alarm MINUTE EQUAL to Clock MINUTE THEN

 IF Alarm SECOND EQUAL to Clock SECOND THEN

 IF Alarm DAY EQUAL to Clock DAY THEN

 IF Alarm MONTH EQUAL to Clock MONTH THEN

 IF Alarm YEAR EQUAL to Clock YEAR THEN

 OUTPUT alarm message

 OUTPUT audible beep

 DEACTIVE alarm

 END IF

 END IF

 END IF

 END IF

 END IF

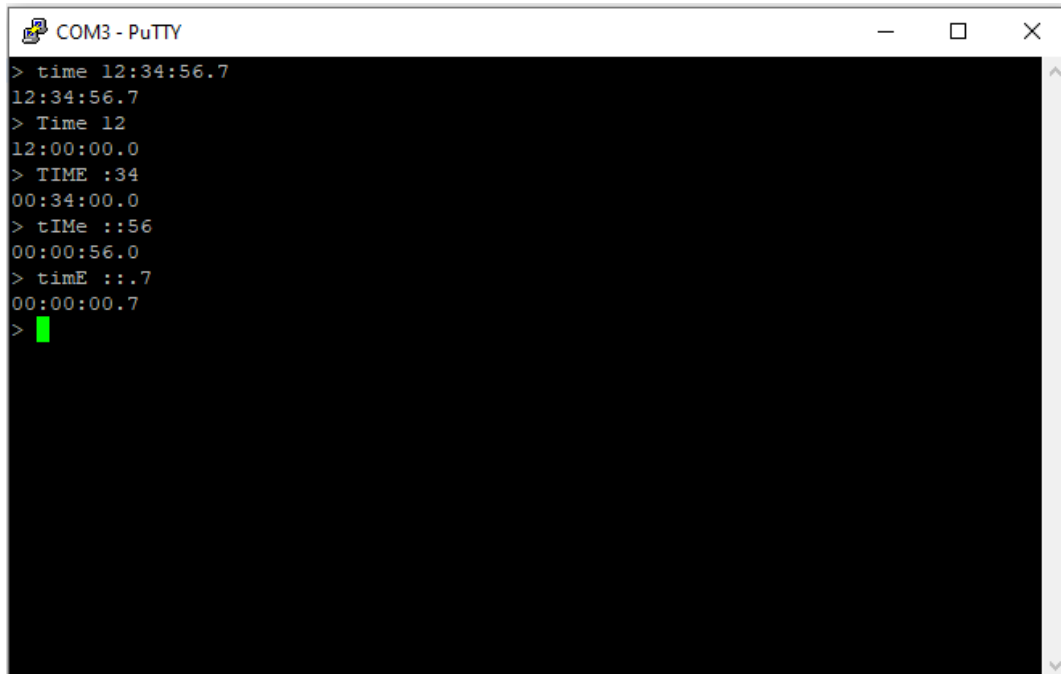
 END IF

END IF

Test:

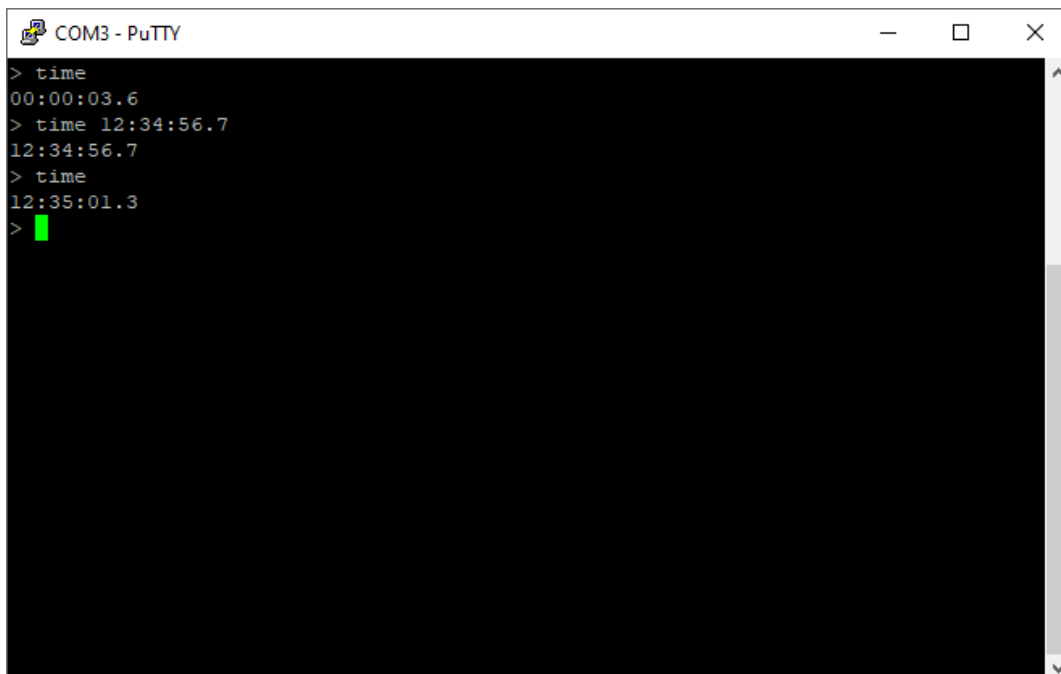
Time setting test:

By input different format of time to test time setting functionality. Time should be all set correctly.



```
COM3 - PuTTY
> time 12:34:56.7
12:34:56.7
> Time 12
12:00:00.0
> TIME :34
00:34:00.0
> tIME ::56
00:00:56.0
> timE ::.7
00:00:00.7
>
```

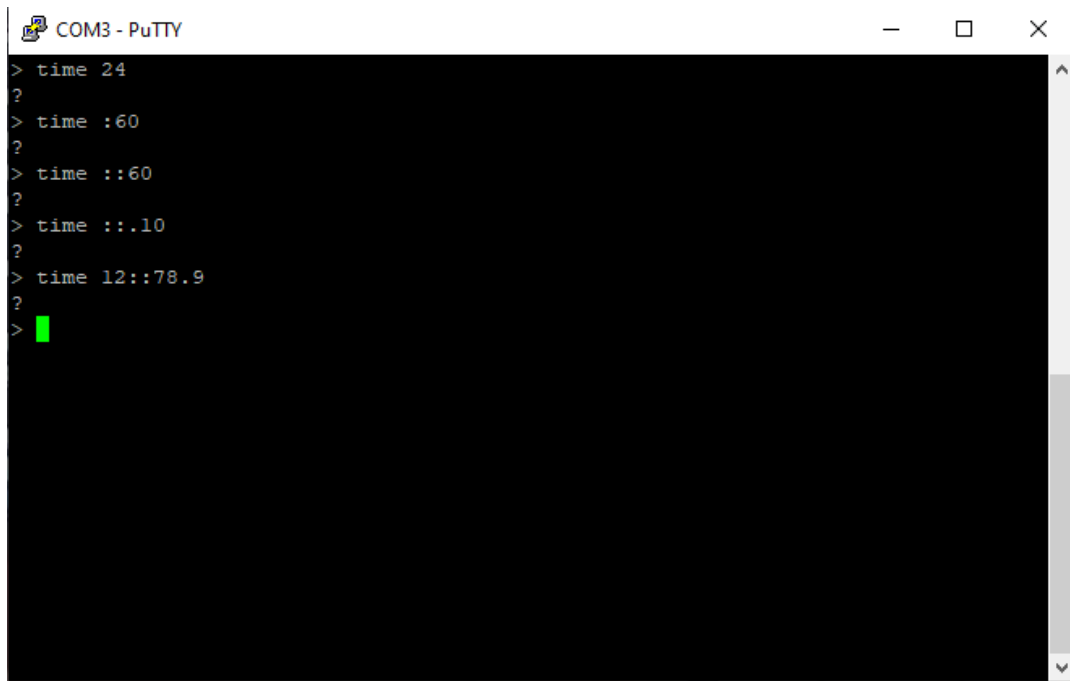
Time reading test:



```
COM3 - PuTTY
> time
00:00:03.6
> time 12:34:56.7
12:34:56.7
> time
12:35:01.3
>
```

Invalid time test:

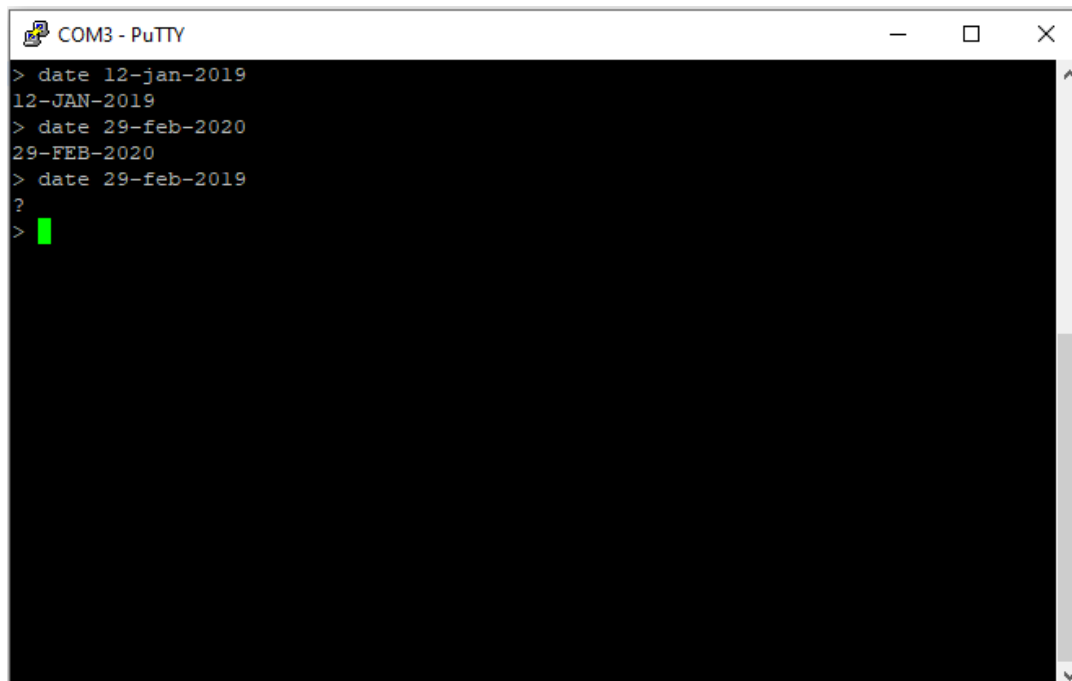
By giving invalid value to test time setting functionality.



```
COM3 - PuTTY
> time 24
?
> time :60
?
> time ::60
?
> time ::.10
?
> time 12::78.9
?
>
```

Date setting test:

By input normal date, leap year Feb 29th and normal year Feb 29th to test date setting functionality. Date should be set correctly except setting Feb 29th in normal year.

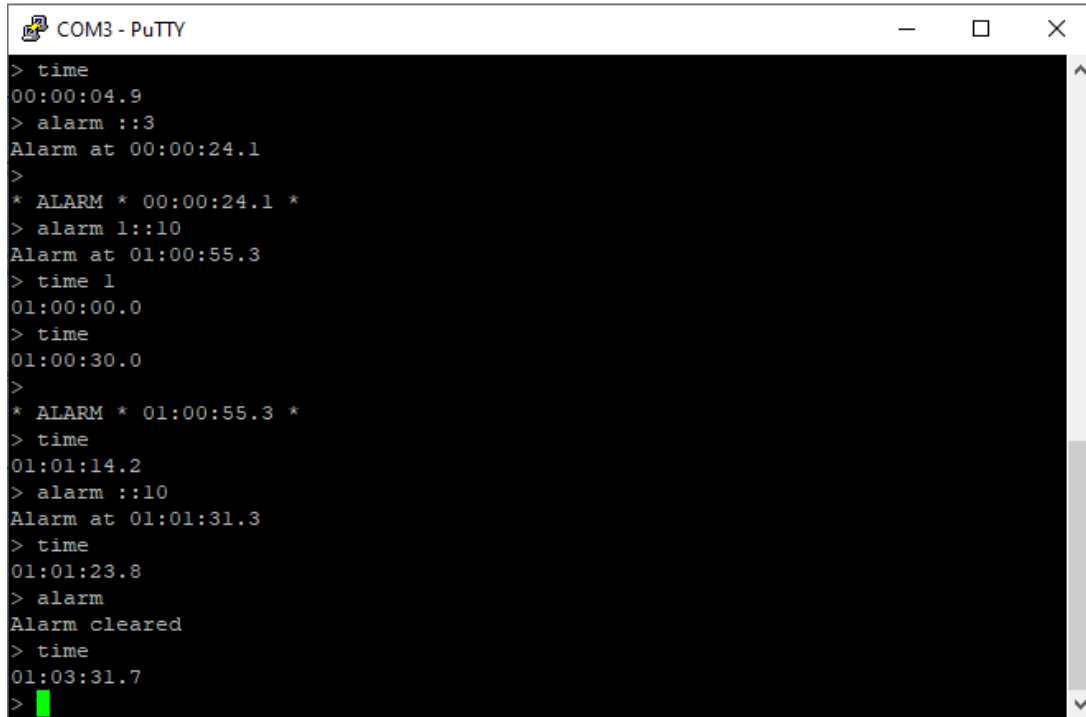


```
COM3 - PuTTY
> date 12-jan-2019
12-JAN-2019
> date 29-feb-2020
29-FEB-2020
> date 29-feb-2019
?
>
```

Alarm testing:

Setting alarm value and alarm should be triggered at correct time.

Setting alarm and then turn off alarm, alarm should not be triggered.



```
COM3 - PuTTY
> time
00:00:04.9
> alarm ::3
Alarm at 00:00:24.1
>
* ALARM * 00:00:24.1 *
> alarm 1::10
Alarm at 01:00:55.3
> time 1
01:00:00.0
> time
01:00:30.0
>
* ALARM * 01:00:55.3 *
> time
01:01:14.2
> alarm ::10
Alarm at 01:01:31.3
> time
01:01:23.8
> alarm
Alarm cleared
> time
01:03:31.7
>
```

Invalid input test:



```
COM3 - PuTTY
> tim
?
>
?
> 12:34:56
?
> alarmm
?
>
```