CoopOS_Demo1_Ultrasonic_ESP8266_Arduino_IDE

1.0

Generated by Doxygen 1.8.11

Contents

1 CoopOS_Simplest-Demo1 / ESP8266-Version / Arduino-IDE

Cooperative multitasking with Esp8266 (Arduino IDE)
(C) 2013-2019 Helmut Weber

Purpose Purpose

License: License

Output Output

Logic Ananlyzer Logic Analyzer

TaskSwitch -Macros TaskSwitch - Macros

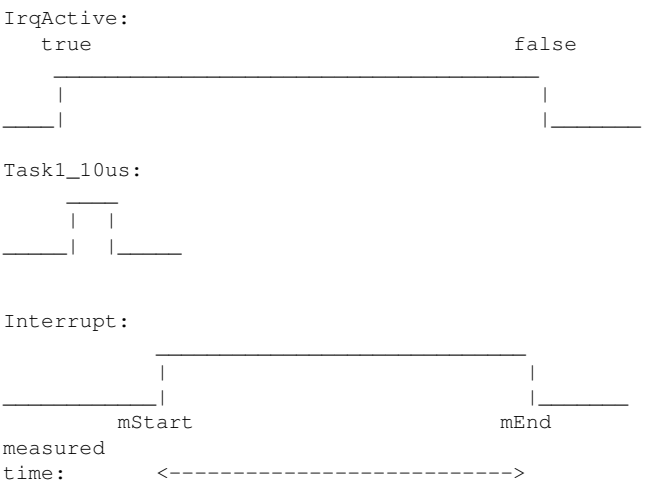
2 License:

3 Purpose

Distance Measuring with Ulttrasonic HC-SR04 with ESP8266 using Arduino IDE

HC-SR04 gets a 10µs Trigger pulse, sends 8 cycles of 40 Khz tone and:

- sets Echo signal HIGH
- sets Echo signal LOW, when Echo arrived:



Concept

A measurement cycle is started from **Task1** :

- set IrqActive=true;
- sends 10µs Trigger signal
- waits for IrqActive=LOW OR Timeout
- filters the result:

```
if (counter<100) {                // no timeout
  diff= (double)(mEnd-mStart);
  if (diff<20000) {
    value=value*0.9 + (0.1*diff); // filtering for noise reduction
  }
}
```

and starts again after a delay.

About 30 measurements per second are done.

The time mStart and mEnd are measured with an interrupt routine, which fires at both edges.
I tests the edge(up,down) and remebers mStart, mEnd.

Task2

Prints the results every 500 ms (twice a second) an shows some alarms like

- Timeout (vla==0)
- distances lower than 300 mm (as an example)
- when something changes the measured distance suddenly (like a cat runnung through the echo path ;)

MySer

Spreads serial output to one character every 50 µs and write to the Uart-Fifo directly.
Serial output is a bottleneck of multitasking.
The Scheduler here is called every 3 µs - when serial output is done it is prolonged to 24 µs.
That is acceptable in most programs.

CoopOS does cooperative multitasking written in pure ANSI-C.

- It is *very* fast (here up to 200000 task switcher per second).
- has a small footprint (here 28 bytes per task).
- reliable
- using microseconds, not milliseconds !

Warning

CoopOS is NOT intended to do numbercrunching !
A lot of time (more than 50%) is used to manage itself.
That's not a failure but a feature;

It is not possible to reach more than 200000 taskswitches per second using only a little amount of processor time.

But it is possible to test for instandce a whole port and start - if any pin has changed - a task blocked until then.
You can do it *before* the scheduler is doing its normal work and that can be as fast as an interrupt - but you have all the possibilities>br> of a task instead of an interrupt routine.
That is possible because the Scheduler() is a small function in your source code - not a big library. So it is easy to customize it for your needs!

The goal is to call as many tasks (or parts of it) as possible, run them very often and with reliable timings and low jitter.
Useful examples: Sample and filter values from an AD-Converter, using Rotary Dials, get serial input as commands, working with Shift Registers, running stepper motors ...etc.
This is for all the embedded systems, which do NOTHING most of the time. And that is right for most of them.>br>

Note

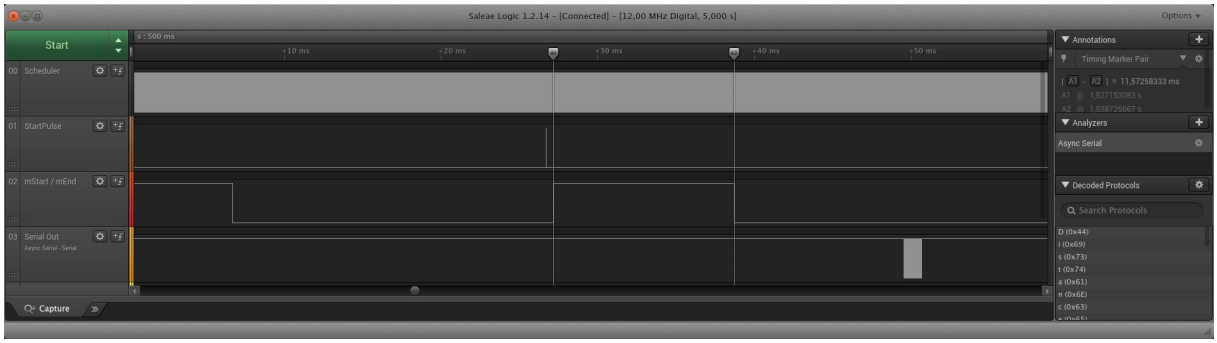
There is an Esp8266 freeRTOS SDK version
Test it - if it may fulfills your timing neeeds!

4 Logic Ananlyzer

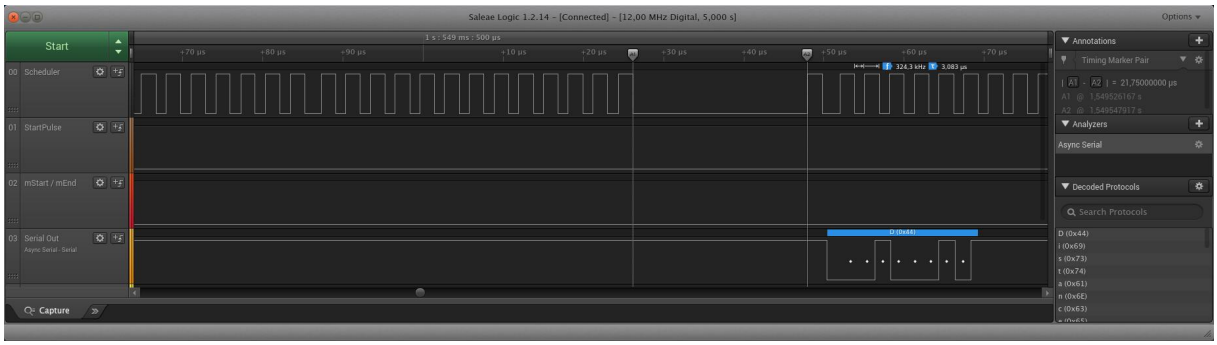
It is highly recommended to use a logic analyzer to test yout CoopOS programs.
Those devices ar not so exdpensiv - really cheap, if you dare to buy chinese rebuilds.

The measurement:

Task switching and serial output



Here we can see how the output of a serial string adds 24 µs.



For analysing a multitasking system you should use a logic analyzer or at leat a scope.

5 Output

Serial output is time consuming. MySerial reduces the time from ms to about 40 µs.
But that means that tasks like Task6 have to take into account that instead of
100 µs cycle time you may have (in rare circumstances) 150 µs.
Do not use serial output or take it.

This is a sample output of the program

```
        OUTPUT

     (C) 2013-2019 Dipl. Phys. H. Weber
     Principles of CoopOS
     3 Tasks running ...
     One Task needs space of 28 Bytes

Init Tasks ...
Init: T_10us function pointer: 0X40202084
Init: Task2 function pointer: 0X402024A4
Init: MySer function pointer: 0X40202280
Init Tasks ready!Distance:  0 1
Distance Alarm:  0 mm

Distance:  1133,3 36
Moving Alarm:    1133,3 mm

Distance:  1299,4 70
Moving Alarm:    1299,4 mm

Distance:  1327,0 104
Distance:  1331,2 139
Distance:  1331,6 169
Distance:  869,1 196
Moving Alarm:    869,1 mm

Distance:  1255,3 230
Moving Alarm:    1255,3 mm
```

```
Distance:  1333,5 678
Distance:  1333,9 712
Distance:  1333,9 747
Distance:  1334,4 782
Distance:  1335,1 816
Distance:  1334,6 850
Distance:  1334,7 884
Distance:  1098,3 918
Moving Alarm:    1098,3 mm

Distance:  930,9 951
Moving Alarm:    930,9 mm

Distance:  1272,3 986
Moving Alarm:    1272,3 mm

Distance:  1324,0 1.020
Distance:  1331,2 1.054
Distance:  1333,2 1.089
Distance:  1333,9 1.124
Distance:  1334,4 1.158
Distance:  1334,6 1.192
```

6 TaskSwitch -Macros

- taskBegin()
  All tasks maust have this before while(1)-loop !!!
- taskEnd()
  All tasks maust have this after while(1)-loop !!!
- taskSwitch()
  Back to Scheduler
- taskDelay(n)
  Back to Scheduler and stop this tas for n microseconds
- taskBlock(n)
  Back to Scheduler and block the task number ID=n until resumed
- taskResume(n)
  Back to Scheduler and set the task number ID=n to READY

7 Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

8 File Index

8.1 File List

Here is a list of all documented files with brief descriptions:

9 Data Structure Documentation

9.1 mySerial Class Reference

Public Member Functions

- void **setSerial** (Stream ∗streamObject)
- void **write** (byte b)
- void **toSer** (char c)
- void **write** (char c)
- void **println** ()
- void **print** (char ∗str)
- void **println** (char ∗str)
- void **print** (unsigned int i)
- void **println** (unsigned int i)
- void **print** (uint8_t i)
- void **print** (uint8_t i, uint8_t n)
- void **print** (unsigned int i, uint8_t n)
- void **println** (unsigned int i, int n)
- void **println** (uint8_t i)
- void **print** (int i)
- void **println** (int i)
- void **print** (unsigned long i)
- void **println** (unsigned long i)
- void **print** (long i)
- void **println** (long i)
- void **print** (uint64_t i)
- void **println** (uint64_t i)
- void **print** (float i)
- void **println** (float i)
- void **print** (double i)
- void **println** (double i)
- char **read** ()
- bool **available** ()
- void **flush** ()

9.1.1 Detailed Description

Definition at line 120 of file MySerial.h.

The documentation for this class was generated from the following file:

- MySerial.h

9.2 task Struct Reference

Data Fields

- unsigned int **ID**
- const char * **Name**
- volatile TState **State**
- uint32_t **LastCalled**
- uint32_t **Delay**
- uint32_t(* **Func**)()
- int **Priority**

9.2.1 Detailed Description

Definition at line 238 of file Esp8266_CoopOS_Demo1_Ultraschall.ino.

The documentation for this struct was generated from the following file:

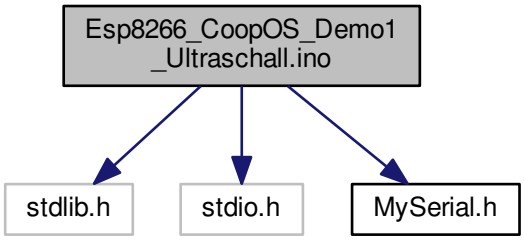
- Esp8266_CoopOS_Demo1_Ultraschall.ino

10 File Documentation

10.1 Esp8266_CoopOS_Demo1_Ultraschall.ino File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "MySerial.h"
```

Include dependency graph for Esp8266_CoopOS_Demo1_Ultraschall.ino:



Data Structures

- struct task

Macros

- #define **MAXTASKS** 10
- #define **GPIO_IN** ((volatile uint32_t∗) 0x60000318)
- #define **taskBegin**()
- #define **taskEnd**()
- #define **taskSwitch**() _mark = __LINE__; return(0) ; case __LINE__:;
- #define **taskDelay**(VAL) _mark = __LINE__; return(VAL) ; case __LINE__:;
- #define **taskBlock**(ID) _mark = __LINE__; Tasks[ID].State = BLOCKED;
- #define **taskResume**(ID) Tasks[ID].State = READY; Tasks[ID].Delay = 0;

Typedefs

- typedef unsigned long long **uint64_t**

Enumerations

- enum **TState** { **READY**, **DELAYED**, **BLOCKED** }

Functions

- int **InitTask** (const char ∗name, uint32_t(∗func)(void), int Priority)
- void **Scheduler** ()
- uint32_t **Task_10us** ()
- uint32_t **Task2** ()
- uint32_t **MySer_Task** ()
- void **IRQ_Echo** ()
- void **setup** ()
- void **loop** ()

Variables

- int **nTasks**
- int **thisID**
- unsigned long **SchedulerCalls**
- unsigned long **T1Counter**
- unsigned long **T2Counter**
- uint32_t **Count6**
- int **interruptPin** =0
- int **T_10us**
- volatile uint32_t **mStart**
- volatile uint32_t **mEnd**
- volatile bool **IrqActive** =false
- uint64_t **IrqCount**
- double **value**
- double **diff**
- struct task **Tasks** [MAXTASKS]

10.1.1 Macro Definition Documentation

10.1.1.1 #define taskBegin( )

Value:

```
static int _mark = 0;                                                   \
  switch (_mark) {                                                      \
  case 0:
```

Definition at line 217 of file Esp8266_CoopOS_Demo1_Ultraschall.ino.

10.1.1.2 #define taskEnd( )

Value:

```
_mark = 0;    return -1;                                                \
}
```

Definition at line 223 of file Esp8266_CoopOS_Demo1_Ultraschall.ino.