MTD2A_timer

MTD2A: Model **Train Detection And Action – arduino library** https://github.com/MTD2A/MTD2A
Jørgen Bo Madsen / V1.2 / 09-10-2025

MTD2A_timer is an easy-to-use, advanced and functional C++ class for non-blocking time control.
MTD2A supports parallel processing and asynchronous execution.

The class is among a number of logical building blocks that solve different functions.
Common to all building blocks are:
• They support a wide range of input sensors and output devices
• Are simple to use to build complex solutions with few commands
• They operate non-blocking, process-oriented and state-driven
• Offers extensive control and troubleshooting information
• Thoroughly documented with examples

Table of contents

Feature Description

MTD2A_binary_input process consists of 3 functions:

1. `MTD2A_timer object_name (”object_name” , CountDownMS );`
2. `object_name.timer ( { RESET_TIMER | START_TIMER | PAUSE_TIMER | STOP_TIMER }, countDownMS} );` Called in `void setup ();`
3. `MTD2A_loop_execute ();` Called as the last instruction in `void loop ();`

The first argument uses the default value and the function can be called with none and up to 2 arguments.

```
MTD2A_timer object_name;
MTD2A_timer object_name (”Object_name”);
MTD2A_timer object_name (”Object_name”, countDownMS);
Default:                    (”Object  name”, 0                    );
```

Example of event management process

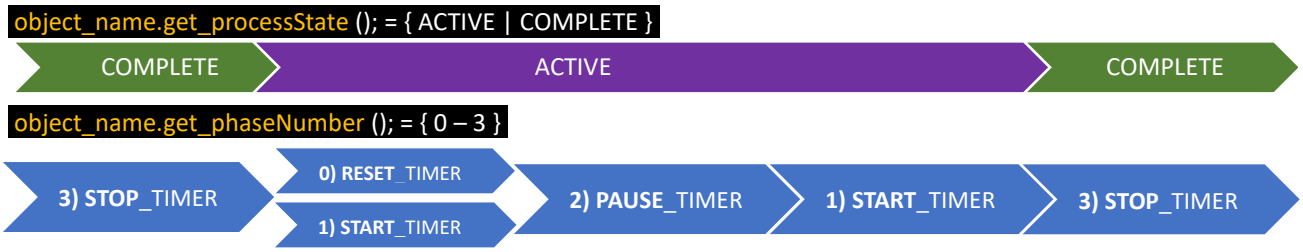
```
// Event if-condition controlled. Recommended for simple solutions
// Read infrared sensor. Short detection: One LED blink.
// continuously detection: continuously LED blink
// Suitable for Unpredictable process flows such as object detection with a sensor
if (IR_sensor.get_processState() == ACTIVE) {
  Serial.println ("Object detected");
  if (timer_GL1.get_processState() == COMPLETE) {
    timer_GL1.timer (START_TIMER, 1000); // 1   second
    green_LED_1.activate (500);          // 0,5 second
  }
}
```

More examples and youtube demo video:

https://github.com/MTD2A/MTD2A/tree/main/examples
DEMO video: https://youtu.be/UU4k4_8GWfM

Process phases

Depending on the current configuration, the process is carried out in between 2 and multiple phases with pauses.



3. The initial phase when the program starts STOP_TIMER.
0. When function object_name.timer (RESET_TIMER); Is called. Can occur at any time.
1. When function object_name.timer (START_TIMER); Is called. May occur several times after each pause.
2. When function object_name.timer (PAUSE_TIMER); Is called. May occur several times after start and reset.
3. When function object_name.timer (STOP_TIMER); Is called or when the time has expired. remainTimeMS = 0.

Global Number Constants:
RESET_TIMER, START_TIMER, PAUSE_TIMER og STOP_TIMER

The immediate phase shift can be identified by function: object_name.get_phaseChange (); = { true | false }

Proces status

When transitioning to START_TIMER or RESET_TIMER, ProcessState switches to ACTIVE.
When transitioning to STOP_TIMER, the processState switches to COMPLETE.
ProcessState remains ACTIVE during PAUSE_TIMER.

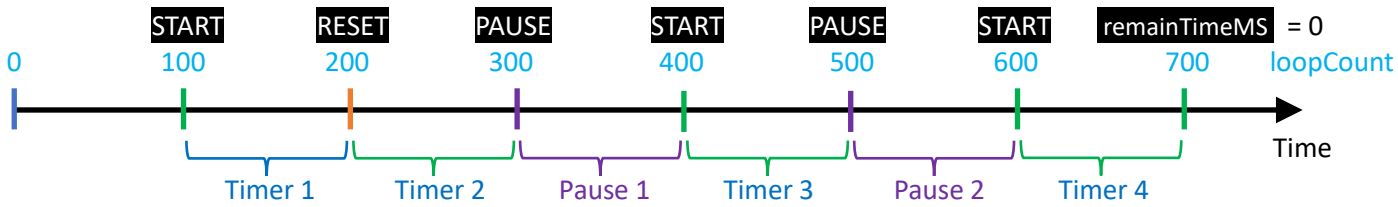
Timing

The time period can be set when the object is instantiated. Subsequently, new time periods can be defined with the functin: object_name.set_countDownMS ({0 - 4294967295}); But only when processState is COMPLETE

Refer to the document MTD2A.PDF and the section "Cadence", "Synchronization" and "Execution speed".

Example of time-controlled process

Timer function is activated at 1000 millisecond intervals (100 loopCount).



Timer 1 deleted
ElapsedTimeMS = Timer 2 + Timer 3 + Timer 4 = approx 3.000 milliseconds.
remainTimeMS = countDownMS – elapsedTimeMS = 0 milliseconds.
PauseTimeMS = Pause 1 + Pause 2 = approx 2.000 milliseconds.

MTD2A_timer object_name ("Timer" , 3000);

```
  // Standard Loop executing time = 10 milliseconds
  if (loopCount ==   0) {  T1.set_debugPrint ();    }
  if (loopCount == 100) {  T1.timer (START_TIMER);  }  // 1 second
  if (loopCount == 200) {  T1.timer (RESET_TIMER);  }  // 2 seconds
  if (loopCount == 300) {  T1.timer (PAUSE_TIMER);  }  // 3 seconds
  if (loopCount == 400) {  T1.timer (START_TIMER);  }  // 4 seconds
  if (loopCount == 500) {  T1.timer (PAUSE_TIMER);  }  // 5 seconds
  if (loopCount == 600) {  T1.timer (START_TIMER);  }  // 6 seconds

  loopCount++;
  if (loopCount == 700) { // 7 seconds
    T1.print_conf ();
    loopCount = 0;
  }
```

IDE Serial Monitor

```
18:04:52.200 -> Timer [1] Start timer
18:04:53.204 -> Timer [0] Reset timer
18:04:54.205 -> Timer [2] Pause timer
18:04:55.216 -> Timer [1] Start timer
18:04:56.216 -> Timer [2] Pause timer
18:04:57.250 -> Timer [1] Start timer
18:04:58.222 -> Timer [3] Stop timer
```

```
MTD2A_timer:
--------------------
  objectName    : Timer
  processState  : COMPLETE
  phaseText     : [3] Stop timer
  debugPrint    : ENABLE
  globalDebugPr: DISABLE
  errorPrint    : DISABLE
  globalErrorPr: ENABLE
  errorNumber   : 0 OK
  countDownMS   : 3000
  remainTimeMS  : 0
  elapsedTimeMS: 3000
  startTimeMS   : 3012
  stopTimeMS    : 8026
  pauseTimeMS   : 2012
  pauseBeginMS  : 6032
  pauseEndMS    : 7038
```

Last measured pause period

Print_conf();

`object_name.print_conf ();`

Maximum time accuracy with Arduino ESP32 and cadence `DELAY_1MS`

```
MTD2A_timer:
--------------------
  objectName    : Timer
  processState  : COMPLETE
  phaseText     : [3] Stop timer
  debugPrint    : ENABLE
  globalDebugPr: DISABLE
  errorPrint    : DISABLE
  globalErrorPr: ENABLE
  errorNumber   : 0 OK
  countDownMS   : 8000
  remainTimeMS  : 6000
  elapsedTimeMS: 2000
  startTimeMS   : 4576
  stopTimeMS    : 9576
  pauseTimeMS   : 3000
  pauseBeginMS  : 6576
  pauseEndMS    : 9576
```

Last measured pause period

If there are multiple breaks during the time period, it shows the total pause time `pauseTimeMS`, but `pauseBeginMS` and `pauseEndMS` only appear for the last pause.

Set and Get Features Overview

Set functions	**Comment**
set_countDownMS ({0 - 4294967295});	Set new count down time in milliseconds
set_debugPrint ({**ENABLE** | DISABLE});	Activate print phase number and text.
set_errorPrint ({**ENABLE** | DISABLE});	Activate error messages.

Get functions	**Comment**
get_processtState (); return bool {ACTIVE | COMPLETE}	Process state
get_phaseChange (); return bool {true | false}	Momentarily phase change (one loop time)
get_phaseNumber (); return uint8_t {0 - 3}	RESET_TIMER = 0, START_TIMER =1, PAUSE_TIMER = 2, STOP_TIMER = 3
get_startTimeMS (); return uint32_t milliseconds	Last START_TIMER
get_stopTimeMS (); return uint32_t milliseconds	STOP_TIMER or remainTimeMS is zero
get_pauseTimeMS (); return uint32_t milliseconds	Acuumulated pause time (sum of all pause periods). Zero if no pause initiated
get_remainTimeMS (); return uint32_t milliseconds	Remining time since first start
get_elapsedTimeMS (); return uint32_t milliseconds	elapsed time since first start
get_reset_error (); return uint8_t {0-255}	Get error/warning number and reset number: Error [1 – 127] warning [128 – 255]

Operator overloading	**Function**
object_name_1 == object_name_2	bool processState_1 == processState_2
object_name_1 != object_name_2	bool processState_1 != processState_2
object_name_1 > object_name_2	bool processState_1 = ACTIVE & processState_2 = COMPLETE
object_name_1 < object_name_2	bool processState_1 = COMPLETE & processState_2 = ACTIVE
object_name_1 >> object_name_2	bool stopTimeMS_1 > stopTimeMS_2
object_name_1 << object_name_2	bool stopTimeMS_1 < stopTimeMS_2