

MTD2A_binary_output

MTD2A: Model Train Detection And Action – arduino library <https://github.com/MTD2A/MTD2A>

Jørgen Bo Madsen / V1.2 / 18-05-2025

MTD2A_binary_output er en avanceret og funktionel C++ klasse til tidsstyret håndtering af output til relæer, LED'er med mere samt programmet selv.

Klassen er blandt en række logiske byggeklodser, der løser forskellig funktioner. Fælles for dem alle:

- Understøtter en bred vifte af inputsensorer og outputenheder
- Enkel at bruge til at bygge komplekse løsninger
- Ikke-blokerende, begivenhedsdrevet, enkel og effektiv tilstandsmaskine

Indholdsfortegnelse

MTD2A_binary_output.....	1
Funktionsbeskrivelse	1
Initialisering og aktivering.....	3
Øvrige funktioner.....	4

Funktionsbeskrivelse

MTD2A_binary_output processen består af fire funktioner:

1. `MTD2A_binary_output object_name`
`("object_name", outputTimeMS, beginTimeMS, endTimeMS, { binary | PWM }, pinBeginValue, pinEndValue);`
2. `object_name.initialize (pinNumber, startPinValue);`
Kaldes i `void setup ();` og efter `Serial.begin ("Hastighed");`
3. `object_name.activate ();` Aktiver en gang og virker først igen når processen er afsluttet (`pending`)
4. `object_name.loop_fast ();`
Kaldes som det sidste i `void loop ();` i den absolut sidste linje tilføjes `delay (10);`

Alle funktioner benytter default værdier og kan derfor kaldes med ingen og op til max antal parametre. Dog skal parameret angives i stigende rækkefølge startende fra den første. Se eksempl herunder:

```
MTD2A_binary_input object_name ();  
MTD2A_binary_input object_name  
("object_name");  
("object_name", outputTimeMS);  
("object_name", outputTimeMS, beginTimeMS);  
("object_name", outputTimeMS, beginTimeMS, endTimeMS);  
("object_name", outputTimeMS, beginTimeMS, endTimeMS, pinOutputMode);  
("object_name", outputTimeMS, beginTimeMS, endTimeMS, pinOutputMode, pinBeginVlaue);  
("object_name", outputTimeMS, beginTimeMS, endTimeMS, pinOutputMode, pinBeginVlaue, pinEndValue);
```

Defaults:

```
("Object name", 0, 0, 0, binary, HIGH, LOW);
```

Eksempel

```
// Read sensor and write phase state information to Arduino IDE serial monitor
#include <MTD2A.h>

MTD2A_binary_output red_LED ("Red LED", 400, 400); // 0.4 sec light, 0.4 sec no light
MTD2A_binary_output green_LED ("Green LED", 300, 700, 0, PWM, 96); // 0,3 / 0.7 sec PWM dimmed

void setup() {
  Serial.begin(9600);
  red_LED.initialize (9); // Output pin 9
  green_LED.initialize (10); // Output pin 10
  Serial.println("Two LED blink");
}

void loop() {
  if (red_LED.get_processState() == pending) red_LED.activate();
  if (green_LED.get_processState() == pending) green_LED.activate();

  red_LED.loop_fast();
  green_LED.loop_fast();

  delay (10);
}
```

Flere eksempler og youtube videoer:

<https://github.com/MTD2A/MTD2A/tree/main/examples>

Youtube videoer: På vej...

Proces faser

Afhængig af den aktuelle konfiguration gennemføres processen i mellem 1 og 5 faser.



0. Den initiale fase når programmet starter samt når funktion `reset ();` kaldes.
1. Start forsinkelse. Hvis sat til 0 eller ikke defineret springes fasen over.
2. Output tidsperiode. Starter med `pinBeginValue` og slutter med `pinEndValue`. Hvis sat til 0 eller ikke defineret springes fasen over.
3. Slut forsinkelse. Hvis sat til 0 eller ikke defineret springes fasen over.
4. Processen er afsluttet `complete` og klar til ny aktivering `object_name.activate ();`

Globale nummerkonstanter: `resetPhase`, `beginPhase`, `outputPhase`, `endPhase`, `completePhase`

Der er to funktioner der kan oplyse hvilken fase processen befinder sig i:

1. `object_name.get_phaseChange (); = { true | false }`
2. `object_name.get_phaseNumber (); = { 0 - 4 }`

Proces status

Ved overgang til `beginPhase`, `outputPhase` eller `endPhase` skifter ProcessState til `active`.

Ved overgang til `completePhase` skifter processState til `complete`.

ProcessState kan aflæses med funktionen `object_name.get_processState (); = { active | complete }`

Timing

Tidstagningen er ikke præcis. Dels tager det ekstra tid at gennemløbe koden, og dels tilføjes tiden fra et ekstra loop gennemløb på 1- 10 millisekunder. Skal tidstagningen være mere præcis, skal det modregnes i de tider der angives når objektet oprettes. Fx ønsket tid: 2.000 – modregnet tid ca. 1.985 millisekunder.

Initialisering og aktivering

Output skrives til det digitale benforbindelsesnummer, der er specificeret i `object_name.initialize (pinNumber);`
Kaldes funktionen ikke, bliver der ikke skrevet til benforbindelsen `pinWrite = disable` og `pinNumber = 255`.

Hvis benforbindelsesnummer er initialiseret korrekt med ovenstående funktion, er det muligt løbende at styre om der skal skrives til benforbindelsen ej med funktionen:

```
object_name.set_pinWrite ( {enable | disable} );
```

Det er også muligt at skrive direkte til benforbindelsen med funktionen:

```
object_name.set_setPinValue ( setPinValue ); binary {HIGH | LOW} / PWM {0-255}
```

Som udgangspunkt er benforbindelsen udefineret. Se [Digital Pins | Arduino Documentation](#)

Det anbefales at definere startværdien for benforbindelsen med funktionen:

```
object_name.initialize ( pinNumber, StartPinValue ); binary {HIGH | LOW} / PWM {0-255}
```

Processen aktiveres med funktionen: `object_name.activate ();` Dermed skifter `procesState` til `active`
Efterfølgende aktivering har ingen effekt, så længe processen er aktiv. Så snart `procesState` skifter til `complete` kan processen aktiveres igen.

Processen kan til hver en tid nulstilles med funktionen: `object_name.reset ();`

Alle styrings og process variable og gør klar til ny start. Alle funktionskonfigurerede variable og standard værdier bibeholdes. Procesfasen skifter til `resetPhase`

Øvrige funktioner

Set functions	Comment
set_pinWrite ({enable disable});	Enable or disable pin writing
set_setPinValue ({binary {HIGH LOW} / PWM {0-255} })	Write directly to output pin (if enabled).
set_debugPrint ({enable disable});	Activate print phase number and text

Fælles for alle set-funktioner er en ekstra parameter: LoopFastOnce = {enable | **disable**} disable er default.

Get functions	Comment
get_processtState (); return bool {active pending}	Procedure process state.
get_pinWrite (); return bool {enable disable}	Write to pin is enabled or disabled
get_phaseChange (); return bool {true false}	Momentarily phase change (one loop time)
get_phaseNumber (); return uint8_t {0- 4}	Initialize & reset = 0, begin = 1, output = 2, end = 3, complete = 4.
get_setBeginMS(); return uint32_t milliseconds	Start time for begin proces
get_setOutputMS (); return uint32_t milliseconds	Start time for output proces
get_setEndMS (); return uint32_t milliseconds	Start time for end proces
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 = pending
object_name_1 < object_name_2	bool processState_1 = pending & processState_2 = active
object_name_1 >> object_name_2	bool setOutputMS_1 > setOutputMS_2
object_name_1 << object_name_2	bool setOutputMS_1 < setOutputMS_2

object_name.print_conf ();

```
MTD2A_binary_output:
-----
objectName      : LED_1
processState    : Active
phaseText       : [3] End delay
debugPrint      : Disable
errorNumber     : 0 OK
outputTimeMS    : 2000
beginDelayMS    : 2000
endDelayMS      : 2000
pinOutputMode   : PWM
pinBeginValue   : 10
pinEndValue     : 0
pinNumber       : 9
pinWrite        : Enable
startPinValue   : 0
setPinValue     : 0
setBeginMS      : 0
setOutputMS     : 2014
setEndMS        : 4028
```