Document Name	Specification of TIMER Driver (SWS)
Document Author	HEX CLAN
Document Status	Published
Version Release Date	6/11/2023

1) Introduction and functional overview

The proposed AVR Timer Driver is a C language library designed to provide efficient and flexible control over the AVR microcontroller's timer functionality. This SRS document outlines the specifications, functionalities, and features of the AVR Timer Driver.

- **Features**

• Single Compare Unit Counter

• Clear Timer on Compare Match (Auto Reload)

• Glitch-free, Phase Correct Pulse Width Modulator (PWM)

• Frequency Generator

• External Event Counter

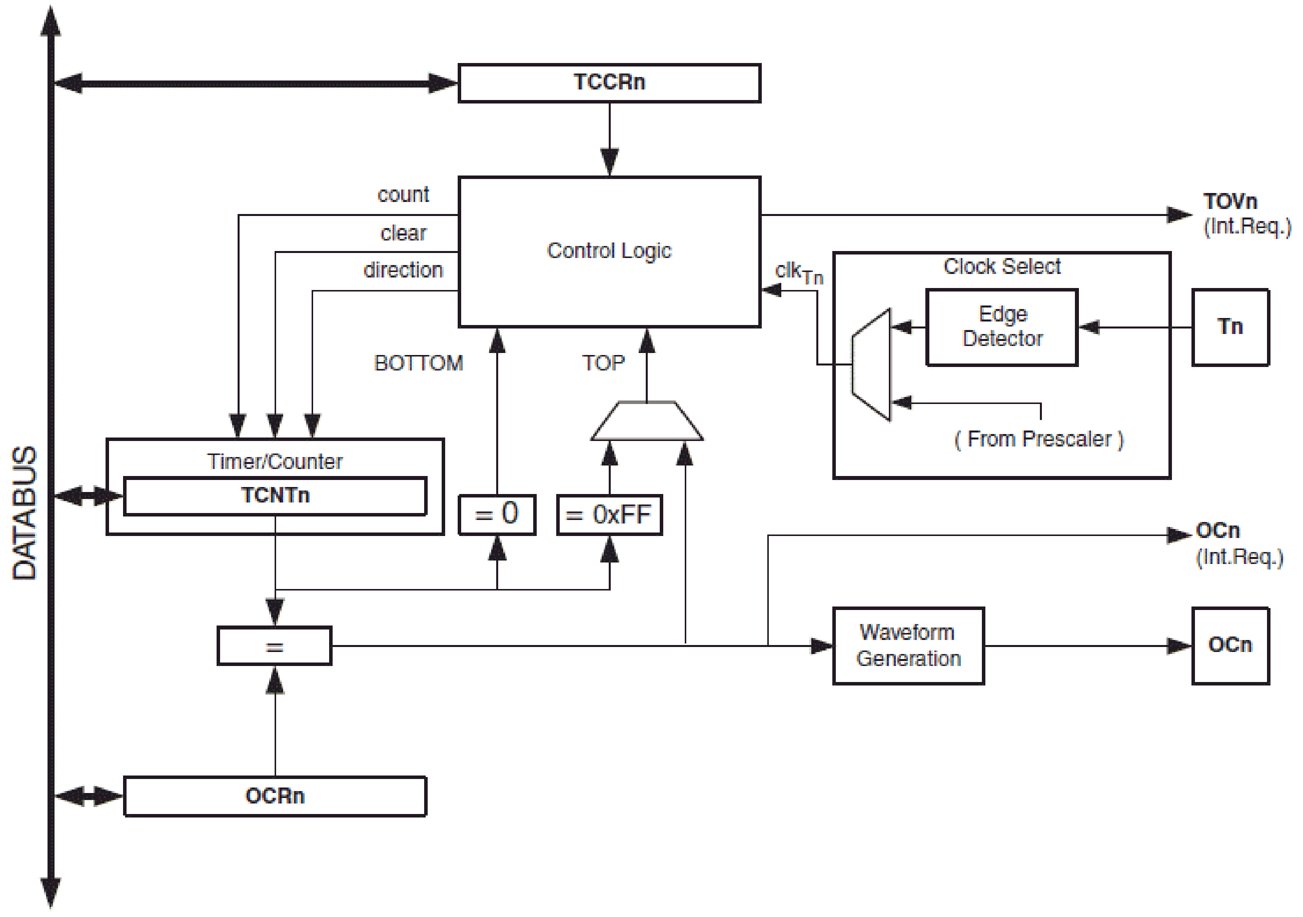
• 10-bit Clock Prescaler

• Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)

- **Overview**

Timer/Counter0 is a general purpose, single compare unit, 8-bit Timer/Counter module. A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 15-1. For the actual placement of I/O pins, refer to “Pinout ATmega32A” on page 10. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “Register Description” on page 86

8-bit Timer/Counter Block Diagram



ss

scope of document:

The AVR Timer Driver aims to provide the following key features:

- Initialization function for timers
- Overflow mode configuration
- CTC mode configuration
- Pulse Width Modulation (PWM) configuration for both Phase Correct and Fast PWM modes
- Input Capture Unit (ICU) functionality

2) Acronyms and abbreviations

Acronyms and abbreviations	**Description**
DIO channel	Represents a single general-purpose digital input/output pin
DIO Port	Represents several DIO channels that are grouped by hardware (typically controlled by one hardware register). Example: Port A (8 bit)
STD-High	Represent the Bit Value High = 1
STD-Low	Represent the Bit Value Low = 0
Physical Level (Input)	Two states possible: LOW/HIGH. A bit value '0' represents a LOW, a bit value '1' represents a HIGH.
Physical Level (Output)	Two states possible: LOW/HIGH. A bit value '0' represents a LOW, a bit value '1' represents a HIGH.
ADC	Analog to Digital Converter
CLCD	Character Liquid Crystal Display
ICU	Input Capture Unit
PWM	Pulse Width Modulation
UART	Universal Asynchronous Recover transmitter
U8	Unsigned Character Data Type of size (8-Bits)
Void	Function does not return any value
(Data -type (Var)*)	The Variable points to the address, of size (Datatype), in data assigned to it.

3) Constraints and assumptions

3.1- Limitations: No limitations

3.2- Applicability to car domains: No restrictions

4) Dependencies to other modules

In a microcontroller system, timers can have dependencies on other modules or components, as they are often used to coordinate various tasks and activities. The specific dependencies may vary depending on the application and how timers are used some common dependencies of timers on other modules:

- Clock Source:
- Interrupt Controller
- I/O Ports
- Compare and Capture Units

5) Functional specification

5.1- General Behavior

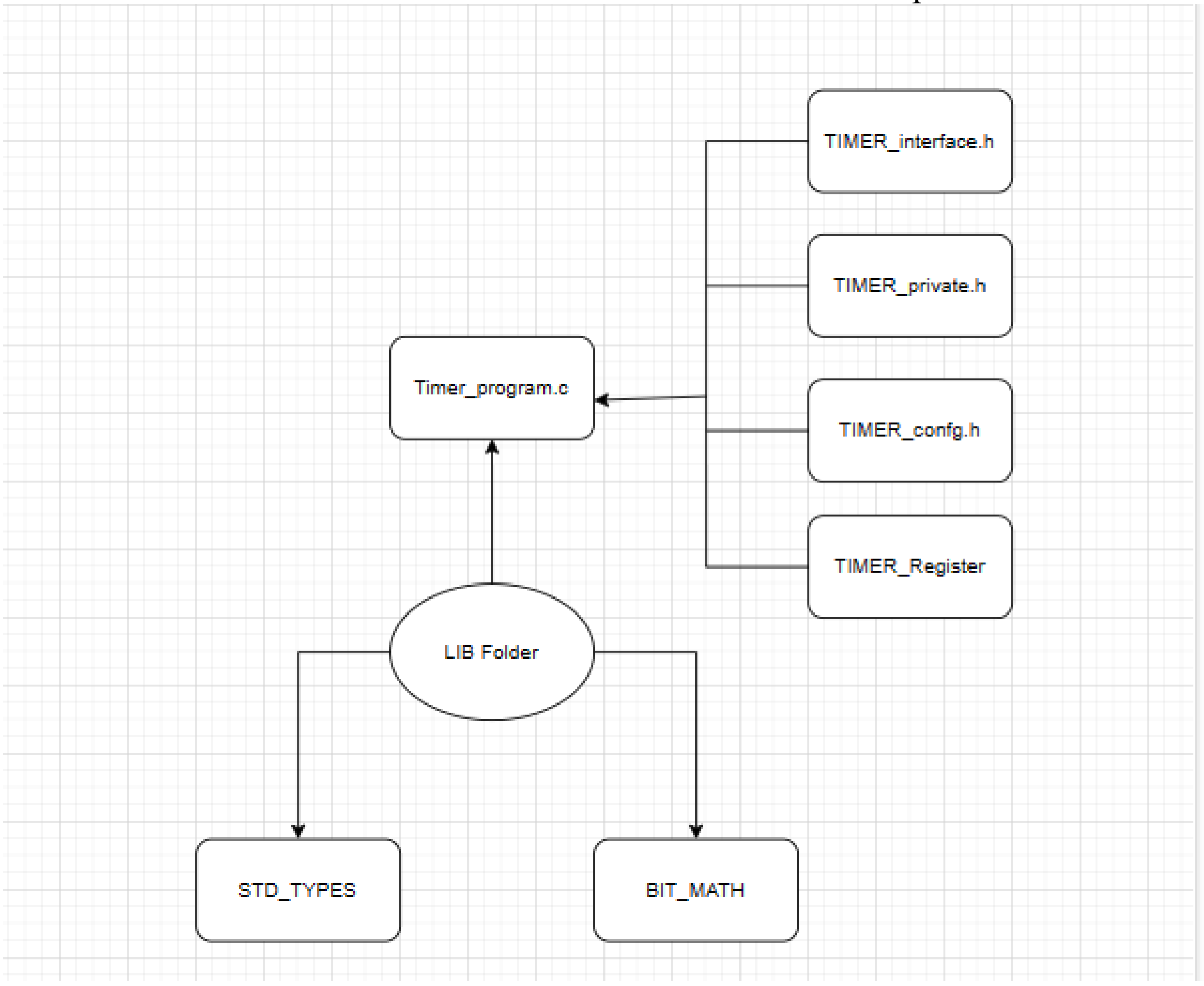
define how the timer can be used and interact with other modules and peripherals in the system. These specifications describe the timer's capabilities and how it can be configured and controlled.

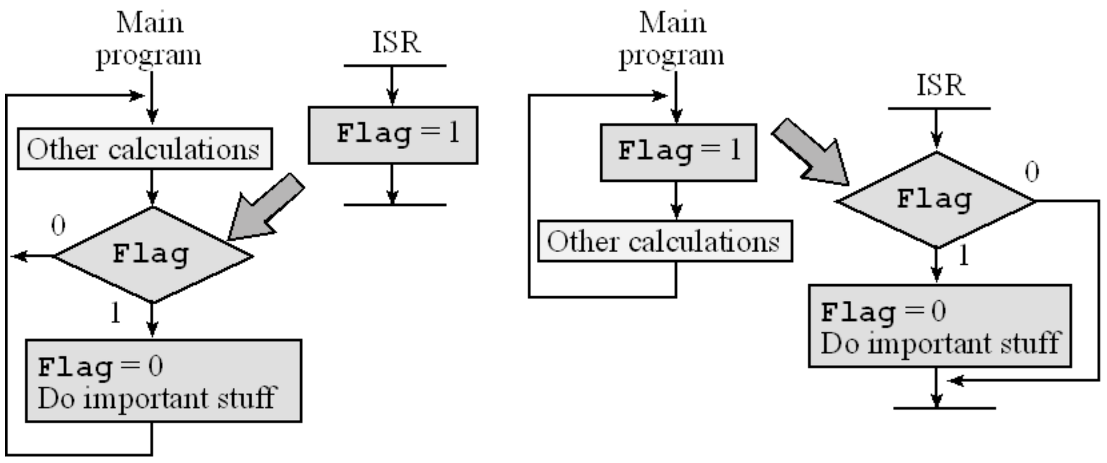
5.1.1- Background & Rationale

Timer drivers abstract the low-level hardware details and complexities of timer configuration, making it more convenient for developers to work with timers in their applications.

5.1.2- Requirements.

A timer driver, or timer library, typically serves as a software abstraction layer that simplifies the usage of timers on a microcontroller.


TIMER_interface.h
TIMER_private.h
Timer_program.c
TIMER_confg.h
TIMER_Register
LIB Folder
STD_TYPES
BIT_MATH



5.2- TIMER write service

5.2.1- Background & Rationale

The timer driver used to measure or control time intervals in a computer system or electronic device. The specific aim of a timer driver can vary depending on the context and the application it is used for

5.2.2- Requirements

[SWS_TIMER0_00064] The TIMER module's write functions shall work for "initialize," is a common convention in programming for a function that is responsible for setting up or initializing the state.

[SWS_ TIMER0_00070] If a TIMER write function is used for a feature where a hardware timer or counter can be configured to generate an output or trigger an action when its count value matches a predefined compare value. This feature is typically used for tasks like generating PWM (Pulse Width Modulation) signals, generating periodic interrupts, or triggering specific actions at precise time intervals.

[SWS_ TIMER0_00109] refers to the process of configuring and setting the initial value of a timer.

5.2.2.1- TIMER channel write service

[SWS_Dio_00006] The DIO-Write Pin Direction

→the function shall set the level of a single DIO channel to Input or Output.

[SWS_Dio_00007] The DIO-Write Pin Value

→ the function shall set the level of a single DIO channel to STD-High or STD-Low.

5.2.2.2- DIO port write service

[SWS_Dio_00009] The DIO-Write Port Direction

→the function shall set simultaneously set the levels of all channels to Input or Output.

[SWS_Dio_00010] The DIO-Write Port Value

→the function shall set simultaneously set the levels of all channels to STD-High or STD-Low.

5.3- *DIO Read Service*

5.3.1- Background & Rationale

The DIO Driver provides services to transfer data from the microcontroller's pins

5.3.2- Requirements

[SWS_Dio_00065] The Dio module's write functions shall work on input and output channels.

5.3.2.1- DIO channel Read service

[SWS_Dio_00011] The DIO-Read Pin Value

→the function shall Read the level of a single DIO. A bit value (0) indicates that the corresponding channel is physical STD_LOW, a bit value (1) indicates that the corresponding channel is physical STD_HIGH.

5.4- DIO Toggle

5.4.1- Background & Rationale

The DIO Driver provides services to toggle data from the microcontroller's pins

5.4.2- Requirements

[SWS_Dio_00066] The Dio module's write functions shall work on input and output channels.

5.4.2.1- DIO channel Toggle service

[SWS_Dio_00012] The DIO-Toggle Pin Value

→the function shall Toggle the level of a single DIO. Change the bit value(0) that corresponds physical STD_LOW, to a bit value (1) that the corresponds to physical STD_HIGH.

5.5- Error classification

5.5.1- Development Errors

Type of Error	Related Error Code	Error Value
ErrorState_t	• Invalid channel requested • Invalid port requested	0

5.5.2 Runtime Errors There are no runtime errors.
5.5.3 Transient Faults There are no transient faults.
5.5.4 Production Errors There are no production errors.
5.5.5 Extended Production Errors There are no extended production errors.

6) API specification

6.1 Imported types

In this chapter all types included from the following modules are listed:

Module	Header File	Imported types
LIB	STD_TYPES.h	U8 (typedef)
	STD_TYPES.h	OK (Error State)
	STD_TYPES.h	NOK (Error State)
	STD_TYPES.h	STD_High
	STD_TYPES.h	STD_Low

6.2 Type Definitions

6.2.1- Dio_PortType

Name	PORT(X)_ID	
Kind	Type	
Range	0 → No. of Ports	Shall Cover All DIO-Ports
Description	Numeric ID of DIO-Port	
Available Via	DIO_Interface.h	

[SWS_Dio_00018] Parameters of type PORT(X)_ID contain the numeric ID of a DIO port.
[SWS_Dio_00181] The mapping of ID is implementation specific but not configurable.
[SWS_Dio_00020] For parameter values of type PORT(X)_ID, the user shall use the symbolic names provided by the configuration description.

6.2.2- Dio_ChannelType

Name	PIN(X)
Kind	Type

Range	0 → No. of Pin	Shall Cover All DIO-Pin in a Port
Description	Numeric ID of each DIO-Pin in each Port	
Available Via	DIO_Interface.h	

[SWS_Dio_00015] Parameters of type PIN(X) contain the numeric ID of a DIO channel.
[SWS_Dio_00180] The mapping of the ID is implementation specific but not configurable.
[SWS_Dio_00017] ⌈For parameter values of type PIN(X), the Dio's user shall use the symbolic names provided by the configuration description.

6.3- Function definitions

Function Name	TIMER0_voidInit
Syntax	void TIMER0_voidInit(void)
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	None
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	None
Description	initializing the state, resources, and configurations of a program.
Available Via	TIMER_Interface.h

Function Name	TIMER0_voidSetCompMatchValue
Syntax	void TIMER0_voidSetCompMatchValue(u8 Copy_u8Value)
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	u8 Copy_u8Value
Parameters (Out)	None

Parameters (In/Out)	None
Return Value	None
Description	used to configure a timer to trigger an event or interrupt when its counter reaches a specific predetermined value,
Available Via	TIMER_Interface.h

Function Name	TIMER0_voidSetTimerValue
Syntax	void TIMER0_voidSetTimerValue(u8 Copy_u8Value)
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	u8 Copy_u8Value
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	None
Description	refers to the specific time duration This value determines how long the timer will run before it triggers an event or an action.
Available Via	TIMER_Interface.h

Function Name	TIMER0_u8OVFSetCallBack
Syntax	u8 TIMER0_u8OVFSetCallBack(void (*Copy_pvTimer0OVFFunc)(void));
Synch/Asynch	Synchronous
Reentrancy	NOnReentrant
Parameters (In)	void (*Copy_pvTimer0OVFFunc)(void)
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	None

Description	When a timer or counter reaches its maximum value and rolls over to zero (overflow), this event can be detected by hardware or software, and a callback function can be executed to perform a specific action or handle the overflow condition.
Available Via	TIMER_Interface.h

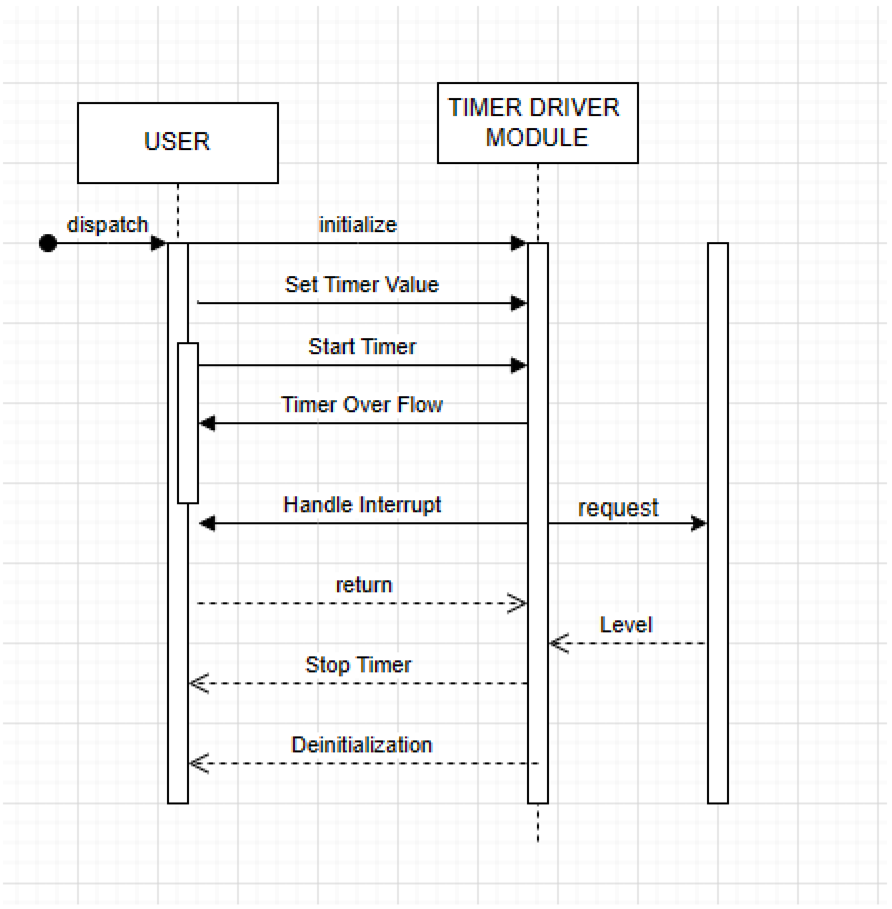
Function Name	TIMER0_u8CompSetCallBack
Syntax	u8 TIMER0_u8CompSetCallBack(void (*Copy_pvTimer0CompFunc)(void));
Synch/Asynch	Synchronous
Reentrancy	NonReentrant
Parameters (In)	void (*Copy_pvTimer0CompFunc)(void)
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	None
Description	This function is used to set up a callback or handler function that is executed when a specific condition is met in a timer.
Available Via	TIMER_Interface.h

Function Name	ICU_voidInit
Syntax	void ICU_voidInit(void);
Synch/Asynch	Synchronous
Reentrancy	Reentrant
Parameters (In)	None
Parameters (Out)	None
Parameters (In/Out)	None
Return Value	None

Description	The ICU function is typically associated with timer units and is often used for various timing and measurement tasks.
Available Via	TIMER_Interface.h

7) Sequence Diagrams

Sequence Diagram for Timer0 Driver Operation


USER
TIMER DRIVER MODULE
dispatch
initialize
Set Timer Value
Start Timer
Timer Over Flow
Handle Interrupt
request
return
Level
Stop Timer
Deinitialization