Firmware development V01

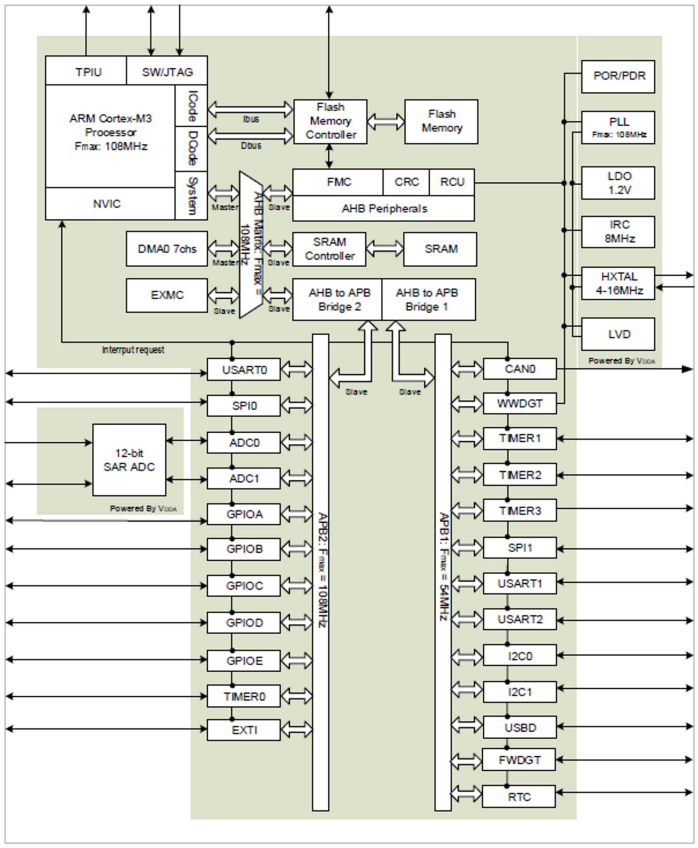
dinsdag 30 november 2021 00:00

The "GD32F103CBT6" is choses for this project.
Main reasoning: mac clk speed is 108MHz twice the speed of the competitors, it also has more ram and flash and additional peripherals. .

GD32F103CBT6
US$ 2.9412
LCSC
Product page

GD32F103...
Datasheet...

Figure 2-1. GD32F103x4/6/8/B block diagram



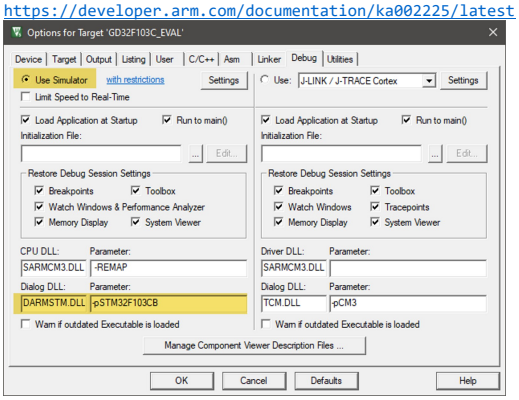
Part Number		F103x CB
Flash (KB)		128
SRAM (KB)		20
Timers	General timer(16-bit)	3 (1~3)
	Advanced timer(16-bit)	1 (0)
	SysTick	1
	Watchdog	2
	RTC	1
Connectivity	USART	3 (0~2)
	I2C	2 (0~1)
	SPI	2 (0~1)
	CAN	1
	USBD	1
GPIO		37
EXMC		0
EXTI		16
ADC	Units	2
	Channels	10

Project name	TK21110_GD32F103CBT6

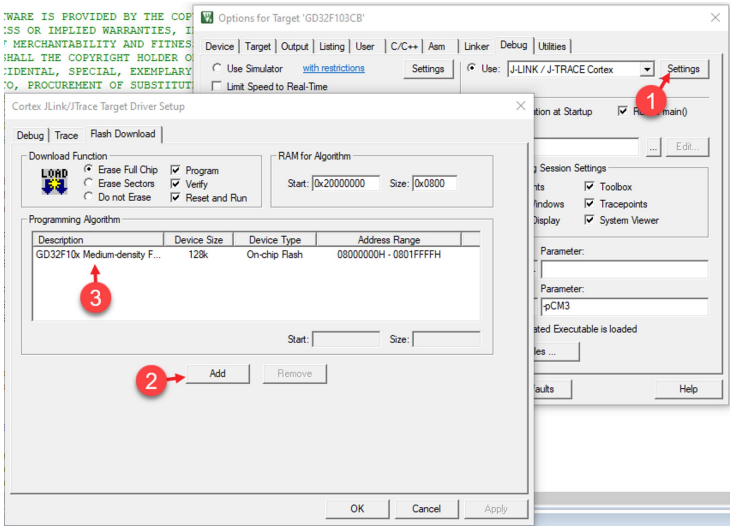
~~NOTE!: The MCU product page does not list the latest files. Use the download site instead to find latest files.~~
~~http://www.gd32mcu.com/en/download/7?kw=GD32F1~~

- [x] When target is changed from example to GD32F103CB, The macro project define "GD32F10X_HD" needs to be changed to GD32F10X_MD. (THIS IS A GUES, but it seems to work and compile without errors)

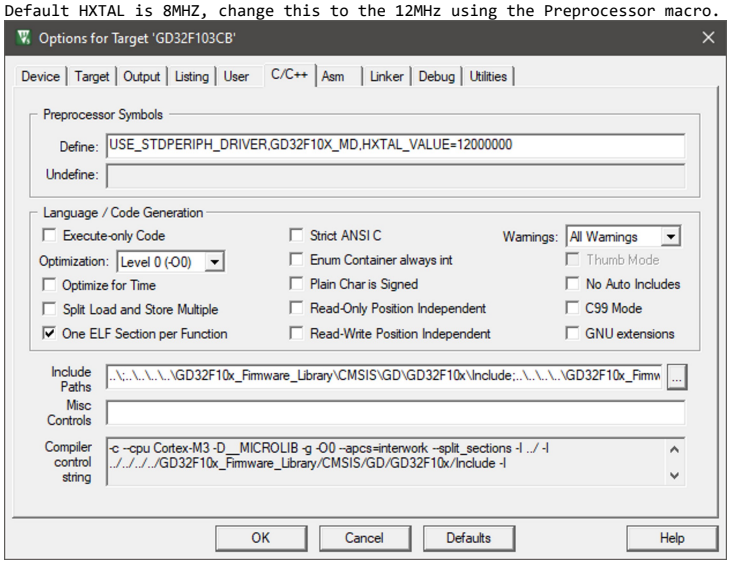
Simulation of the processor and even the peripherals is posible in keil
https://developer.arm.com/documentation/ka002225/latest



*To be able to program the board it is needed to add the programming algorithm.



Default HXTAL is 8MHZ, change this to the 12MHz using the Preprocessor macro.



The CPU clock is configured to the max 108Mhz using External crystal as source:

main.c | startup_gd32f10x_hd.s | system_gd32f10x.c

```
34  */
35
36  /* This file refers the CMSIS standard, some adjustments are made according to GigaDevice chips */
37
38  #include "gd32f10x.h"
39
40  /* system frequency define */
41  #define __IRC8M            (IRC8M_VALUE)            /* internal 8 MHz RC oscillator frequency */
42  #define __HXTAL            (HXTAL_VALUE)            /* high speed crystal oscillator frequency */
43  #define __SYS_OSC_CLK      (__IRC8M)                /* main oscillator frequency */
44
45  /* select a system clock by uncommenting the following line */
46  /* use IRC8M */
47  //#define __SYSTEM_CLOCK_48M_PLL_IRC8M            (uint32_t)(48000000)
48  //#define __SYSTEM_CLOCK_72M_PLL_IRC8M            (uint32_t)(72000000)
49  //#define __SYSTEM_CLOCK_108M_PLL_IRC8M           (uint32_t)(108000000)
50
51  /* use HXTAL (XD series CK_HXTAL = 8M, CL series CK_HXTAL = 25M) */
52  //#define __SYSTEM_CLOCK_HXTAL                    (uint32_t)(__HXTAL)
53  //#define __SYSTEM_CLOCK_24M_PLL_HXTAL            (uint32_t)(24000000)
54  //#define __SYSTEM_CLOCK_36M_PLL_HXTAL            (uint32_t)(36000000)
55  //#define __SYSTEM_CLOCK_48M_PLL_HXTAL            (uint32_t)(48000000)
56  //#define __SYSTEM_CLOCK_56M_PLL_HXTAL            (uint32_t)(56000000)
57  //#define __SYSTEM_CLOCK_72M_PLL_HXTAL            (uint32_t)(72000000)
58  //#define __SYSTEM_CLOCK_96M_PLL_HXTAL            (uint32_t)(96000000)
59  #define __SYSTEM_CLOCK_108M_PLL_HXTAL           (uint32_t)(108000000)
```

```
/* retarget the C library printf function to the USART */
int fputc(int ch, FILE *f)
{
    usart_data_transmit(EVAL_COM0, (uint8_t)ch);
    while(RESET == usart_flag_get(EVAL_COM0, USART_FLAG_TBE));

    return ch;
}
```

Building a working project from the firmware library

GD32F10x_Firmware_Library_V2.1.2.rar
core_cm3.h part of the CMSIS has a problem and references files that do not exsist

Checking what version of CMSIS the firmware package references.
https://www.keil.com/dd2/pack/#!#third-party-download-dialog

GigaDevice::GD32F10x_DFP	latest	2.0.2	GigaDevice GD32F10x Series Device Support and Examples
2.0.2	☑		

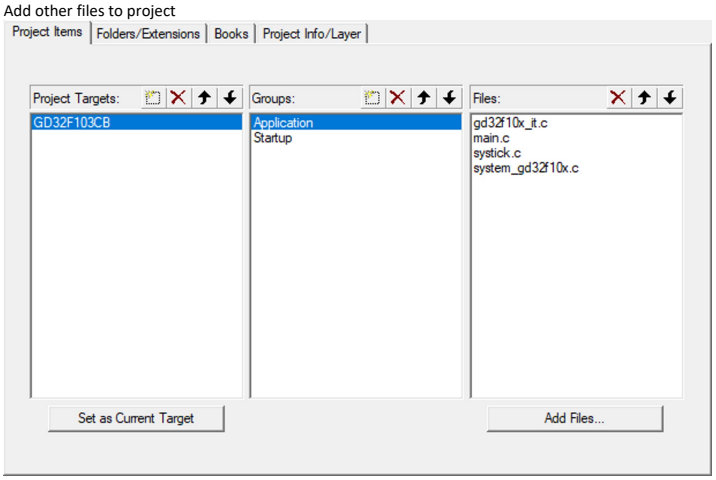
The CMSIS lib is sepperate from the GigaDevice HAL. The GD HAL references and old version of the CMSIS and does not work however keil uses the latest CMSIS (installed via packs) and the projects compile without problems.

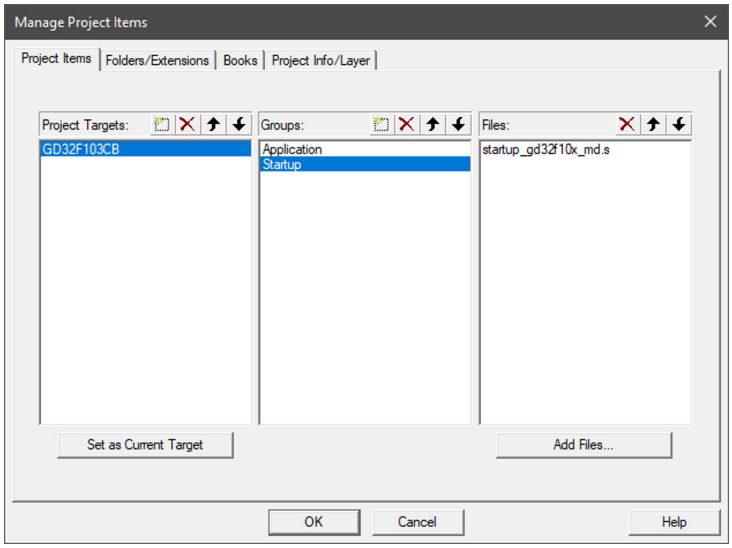
Building working project from scratch

Open keil, go to project -> new project
Select the device "GD32F103CB"
Select required drivers and CMSIS:

Software Component	Sel.	Variant	Version	Description
CMSIS				Cortex Microcontroller Software Interface Components
CORE	☑		5.5.0	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M
DSP	☐	Source	1.9.0-dev	CMSIS-DSP Library for Cortex-M, SC000, and SC300
NN Lib	☐		3.0.0	CMSIS-NN Neural Network Library
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.3	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
GD32F10x_libopt	☑		2.0.1	Configuration file
Startup	☐		2.0.1	System Startup for GigaDevice GD32F101 and GD32F103 High Density Devices
EVAL				
GD32F10x_StdPeripherals				
ADC	☑		2.0.1	Analog-to-digital converter (ADC) driver for GD32F10x Devices
BKP	☐		2.0.1	Backup register(BKP) driver for GD32F10x Devices
CAN	☐		2.0.1	Controller Area Network (CAN) driver for GD32F10x Devices
CRC	☐		2.0.1	Cyclic Redundancy Check (CRC) driver for GD32F10x Devices
DAC	☐		2.0.1	Digital-to-analog converter (DAC) driver for GD32F10x Devices
DBG	☑		2.0.1	Debug (DBG) driver for GD32F10x Devices
DMA	☑		2.0.1	Direct Memory Access (DMA) driver for GD32F10x Devices
ENET	☐		2.0.1	Ethernet(ENET) driver for GD32F10x Devices
EXMC	☐		2.0.1	External memory controller(EXMC) driver for GD32F10x Devices
EXTI	☑		2.0.1	External Interrupt/Event (EXTI) driver for GD32F10x Devices
FMC	☐		2.0.1	Flash Memory Controller (FMC) driver for GD32F10x Devices
FWDGT	☑		2.0.1	Free watchdog timer(FWDGT) driver for GD32F10x Devices
GPIO	☑		2.0.1	General-purpose and Alternate-function I/Os (GPIO) driver for GD32F10x Devices
I2C	☐		2.0.1	Inter-integrated Circuit (I2C) driver for GD32F10x Devices
MISC	☑		2.0.1	MISC driver for GD32F10x Devices
PMU	☑		2.0.1	Power Managment Unit(PMU) driver for GD32F10x Devices
RCU	☑		2.0.1	Reset and Clock Control (RCU) driver for GD32F10x Devices
RTC	☐		2.0.1	Real-time Clock (RTC) driver for GD32F10x Devices
SDIO	☐		2.0.1	Secure digital input/output interface(SDIO) driver for GD32F10x Devices
SPI_I2S	☐		2.0.1	Serial Peripheral Interface / Inter-IC Sound (SPI_I2S) driver for GD32F10x Devices
TIMER	☑		2.0.1	TIMER driver for GD32F10x Devices
USART	☑		2.0.1	Universal Synchronous Asynchronous Receiver Transmitter (USART) driver for GD32F10x Devi...
WWDGT	☑		2.0.1	Window Watchdog Timer (WWDGT) driver for GD32F10x Devices
File System		MDK-Plus	6.14.1	File Access on various storage devices
Graphics		MDK-Plus	6.16.3	User Interface on graphical LCD displays
Network		MDK-Plus	7.15.0	IPv4 Networking using Ethernet or Serial protocols
RTOS		FreeRTOS	10.4.6	FreeRTOS Real Time Kernel

Add other files to project





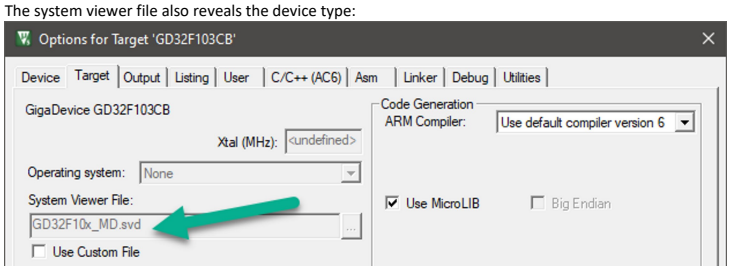
Note how in the pack manager only the startup file for the High density devices can be selected so it is added manualy.
(the user manual states which device type I have)

GigaDevice GD32F10x User Manual

1.5. Device electronic signature

Medium-density devices are GD32F101xx and GD32F103xx microcontrollers which the flash memory density ranges from 16 to 128 Kbytes.
High-density devices are GD32F101xx and GD32F103xx microcontrollers which the flash memory density ranges from 256 to 512 Kbytes.
Extra-density devices are GD32F101xx and GD32F103xx microcontrollers which the flash memory density larger than 512 Kbytes.
Connectivity line devices are GD32F105xx and GD32F107xx microcontrollers.

The system viewer file also reveals the device type:

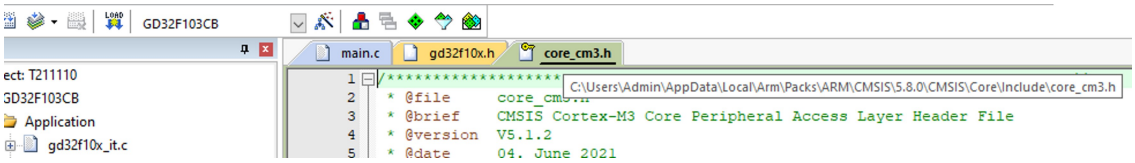


Keil assistant VSCODE

```
#include "gd32f10x.h"    #include errors detected. Please update your includePath. Squiggles are disabled for this translation unit (C:\Users\Admin\Desktop\T211110_Firmware\
```

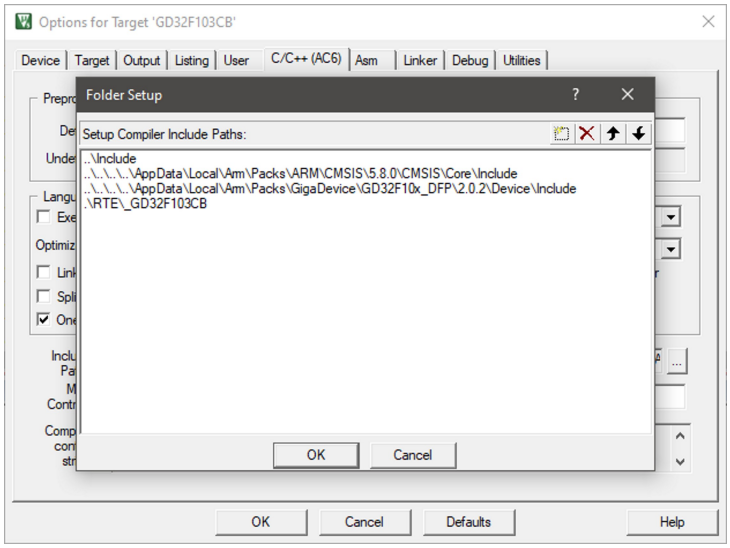
"#include "core_cm3.h":

After opening the project in you will get an include errors. This is because the include dir of the software packs is not imported into the Vscode extension.



Check in Keil were these files are located, and add these include paths to the compiler settings in keil, this way the Vscode extension can find the files.

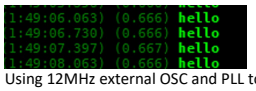
C:\Users\Admin\AppData\Local\Arm\Packs\ARM\CMSIS\5.8.0\CMSIS\Core\Include\core_cm3.h
C:\Users\Admin\AppData\Local\Arm\Packs\GigaDevice\GD32F10x_DFP\2.0.2\Device\Include\gd32f10x_libopt.h
C:\Users\Admin\Desktop\T211110_Firmware\T211110_Firmware_Keil_uVision\RTE_GD32F103CB\RTE_Components.h
C:\Users\Admin\AppData\Local\Arm\Packs\GigaDevice\GD32F10x_DFP\2.0.2\Device\Firmware\Peripherals\inc\gd32f10x_adc.h



Check OSC freq

Using 8MHz internal OSC and PLL to create 108MHZ
This works perfectly

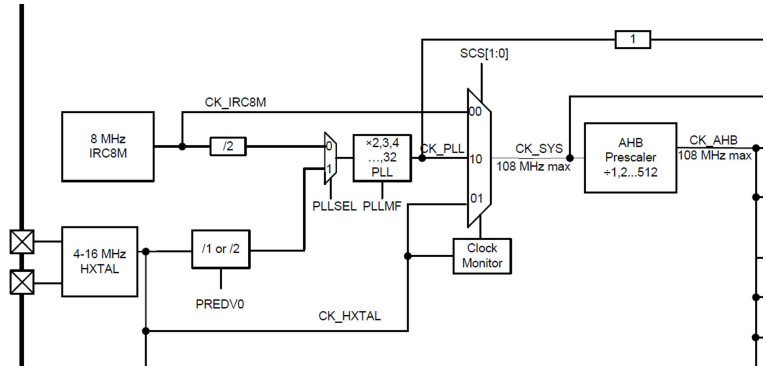
```
#include "gd32f10x.h"

/* system frequency define */
#define __IRC8M            (IRC8M_VALUE)            /* internal 8 MHz RC oscillator frequency */
#define __HXTAL            (HXTAL_VALUE)            /* high speed crystal oscillator frequency */
#define __SYS_OSC_CLK      (__HXTAL)                /* main oscillator frequency */

/* select a system clock by uncommenting the following line */
/* use IRC8M */
//#define __SYSTEM_CLOCK_48M_PLL_IRC8M            (uint32_t)(48000000)
//#define __SYSTEM_CLOCK_72M_PLL_IRC8M            (uint32_t)(72000000)
//#define __SYSTEM_CLOCK_108M_PLL_IRC8M           (uint32_t)(108000000)

/* use HXTAL (XD series CK_HXTAL = 8M, CL series CK_HXTAL = 25M) */
//#define __SYSTEM_CLOCK_HXTAL                    (uint32_t)(__HXTAL)
//#define __SYSTEM_CLOCK_24M_PLL_HXTAL            (uint32_t)(24000000)
//#define __SYSTEM_CLOCK_36M_PLL_HXTAL            (uint32_t)(36000000)
//#define __SYSTEM_CLOCK_48M_PLL_HXTAL            (uint32_t)(48000000)
//#define __SYSTEM_CLOCK_56M_PLL_HXTAL            (uint32_t)(56000000)
//#define __SYSTEM_CLOCK_72M_PLL_HXTAL            (uint32_t)(72000000)
//#define __SYSTEM_CLOCK_96M_PLL_HXTAL            (uint32_t)(96000000)
#define __SYSTEM_CLOCK_108M_PLL_HXTAL           (uint32_t)(108000000)
```



Using 12MHz external OSC and PLL to create 108MHZ

The result is incorrect, probably the code thinks it is using a 8MHz crystal.

I made the incorrect assuption that the code checks the HXTAL_VALUE and calculates the required PLL settings, this is not the case so the code must be modified.

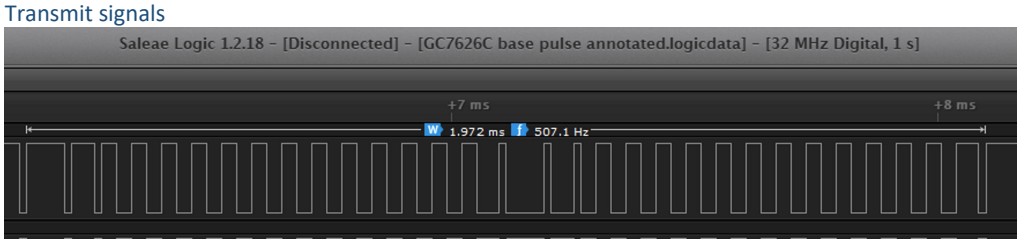


Calculation:

PREDV0
12/1=12
Or
12/2=6

PLLMF
108/12=9
or
108/6=18

Transmit signals



The Full signal is divided in 512 parts.
(1972/512)=3.8516 Us

(1972E-6/512)^-1=259634.8884
~260kHz

Timer prescaler
108000khz/260khz=415.3846

PA15 as output problem

When testing, and trying to output a signal oon the PERIOD pin, PA15 nothing happened.
Looking and the GPIO config register, the output is not configured correctly.

7.3.1. GPIO pin configuration

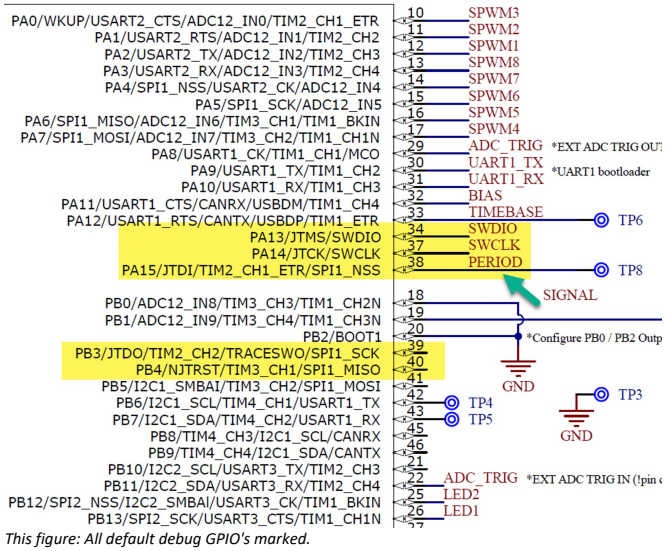
During or just after the reset period, the alternative functions are all inactive and the GPIO ports are configured into the input floating mode that input disabled without Pull-Up (PU)/Pull-Down (PD) resistors. But the JTAG/Serial-Wired Debug pins are in input PU/PD mode after reset:

PA15: JTDI in PU mode.
PA14: JTCK / SWCLK in PD mode.
PA13: JTMS / SWDIO in PU mode.
PB4: NJTRST in PU mode.
PB3: JTDO in Floating mode.

```
63    //Timebase LUT output 1 sample
64    timeBaseInit(); //GPIO PA12 (default)
65
66    //Period LUT output LUTsize = 512 samples
67    periodInit(); // PA15 (defailt JTDI)
```

CTL0	0x00
MD0	0x03
CTL1	0x88833444
CTL15	0x02
MD15	0x00
CTL14	0x02
MD14	0x00

PORTA can't be used while sensor is active



This figure: All default debug GPIO's marked.

So only PA15 causes a problem in the default configuration.

FIX:

7.4.3. JTAG/SWD alternate function remapping

The debug interface signals are mapped on the GPIO ports as shown in table below.

Table 7-2. Debug interface signals

Alternate function	GPIO port
JTMS / SWDIO	PA13
JTCK / SWCLK	PA14
JTDI	PA15
JTDO / TRACESWO	PB3
NJTRST	PB4
TRACECK	PE2
TRACECK0	PE3
TRACECK1	PE4
TRACECK2	PE5
TRACECK3	PE6

To reduce the number of GPIOs used to debug, user can configure SWJ_CFG [2:0] bits in the AFIO_PCF0 to different value. Refer to table below.

Table 7-3. Debug port mapping

SWJ_CFG [2:0]	Available debug ports	SWJ I/O pin assigned				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/S WCLK	PA15/ JTDI	PB3/ JTDO/ TRACE SWO	PB4/ NJTRST
000	Full SWJ (JTAG-DP + SW-DP) (Reset state)	•	•	•	•	•
001	Full SWJ (JTAG-DP + SW-DP) but without NJTRST	•	•	•	•	X
010	JTAG-DP Disabled and SW-DP Enabled	•	•	X	X(1)	X
100	JTAG-DP Disabled and SW-DP Disabled	X	X	X	X	X
Other	Forbidden					

1. Released only if not using asynchronous trace.

```
/*! Configure GPIO for SWD, instead of SWD+JTAG. to release PA15 for GPIO function
*/
void dbgIo_init()
{
    rcu_periph_clock_enable(RCU_AF);
    gpio_pin_remap_config(GPIO_SWJ_SWDPENABLE_REMAP, ENABLE);
}
```

NOTE: AF clock must be enabled first else the register bits can not be changed.

WHOEPS! I disabled SWD and JTAG while testing now the chip can not be flashed again .....
Probably can be fixed using the internal bootloader ~~or option byte programming~~?

Fixed the Chip by booting it inside the internal bootloader (BOOT0 HIGH at reset), connected to the device using the "GD32 All-In-One Programmer" and USB port on the device. Erased the chip using this tool, now all is oke and I only disabled the JTAG this time.
I believe it is also possible to use SWD when the device is booted in the ROM bootloader.

ADC trigger generation.
The ADC is triggered externally, this reasoning behind this is that it makes debugging the signal easy, and no software overhead is needed to trigger the ADC.

The period of the bias signal is 8T (3T low 5T HIGH).
The Bias signal is used to discharge the receiver capacitor in between samples. When the bias signal is low the RX cap is left floating and thus can be charged by the couples TX signals.

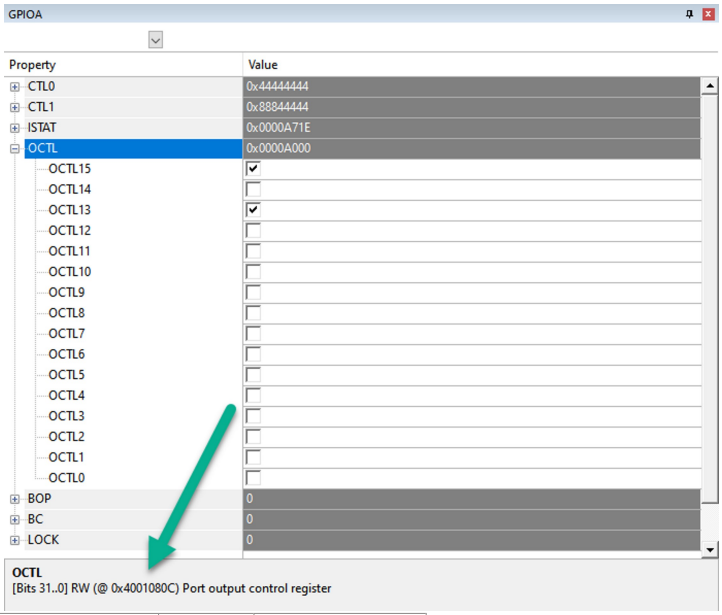
Pink	ADC trigger signal

```
// uint32_t timeBase = (tx1Loc % 2) ? 0 : 1; //Even , odd numers, flips every sample

uint32_t timeBase = ((tx1Loc + 7) % 8) ? 0 : 1; //ADC trigger signal every 8T, Offset by 1 to align ADC trigger (rising edge) on second period when the bias is low.
```

Code used to generate the ADC trigger signal. *timebase is used because this pin was already connected to my scope an thus could be used to check if it was correct.
"+7" is used to align the rising edge of the acd trigger on the start of the second period of the bias signal. So the ADC samples ~in the middle of the signal.

DMA & Timer to output signals without CPU intervention

Checking the memory address that needs to be written by the DMA.



Peripheral_mem_add	0x4001080C	OCTL = **O**utput **C**on**T**ro**L**

DMA circular mode: handly, wel restart transfer automatically in a loop. Usefull for the signal generation, only have to start it one time.

Table 9-3. DMA0 requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
TIMER0	•	TIMER0_CH0	TIMER0_CH1	TIMER0_CH3 TIMER0_TG TIMER0_CMT	TIMER0_UP	TIMER0_CH2	•
TIMER1	TIMER1_CH2	TIMER1_UP	•	•	TIMER1_CH0	•	TIMER1_CH1 TIMER1_CH3
TIMER2	•	TIMER2_CH2	TIMER2_CH3	•	•	TIMER2_CH0	•
			TIMER2_UP			TIMER2_TG	
TIMER3	TIMER3_CH0	•	•	TIMER3_CH1	TIMER3_CH2	•	TIMER3_UP
ADC0	ADC0	•	•	•	•	•	•
SPI/I2S	•	SPI0_RX	SPI0_TX	SPI1/I2S1_RX	SPI1/I2S1_TX	•	•
USART	•	USART2_TX	USART2_RX	USART0_TX	USART0_RX	USART1_RX	USART1_TX
I2C	•	•	•	I2C1_TX	I2C1_RX	I2C0_TX	I2C0_RX

The DMA could be used to handle both the **ADC** signal sample storage and **Sensor signal output generation**.
And even the **UART** could be handled by the DMA, this way the CPU can handle the signal processing.

No timers are used inside the project except the one for the sensor timebase, no special features are used so every timer is viable.
The ADC is fixed however and the UART also.

Sensor input uses:

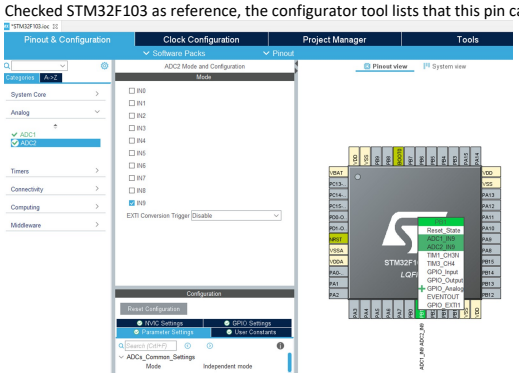
PB1	19	I/O		Default: PB1 Alternate: ADC01_IN9, TIMER2_CH3 Remap: TIMER0_CH2_ON

Not sure what the ADC number is... ADC01_IN9 is used for signal sampling

The GD32F103 has 2 ADC units

Part Number		F103x0
		CB
ADC	Units	2
	Channels	10

Checked STM32F103 as reference, the configurator tool lists that this pin can be connected to both ADC's?



Pin name	Type(1)	I / O Level	Main function(3) (after reset)	Default	Remap
PB1	I/O	-	PB1	ADC12_IN9/ TIM3_CH4(9)	TIM1_CH3N

AHA, The naming convention: ADC1 and ADC2 are combined to ADC12.
The GD32 starts peripheral naming at zero instead of 1 so this explains ADC01_IN9 on PB1.
Back to the DMA.

DMA0 config (GD32F103CB has only one DMA controller DMA0):

Table 9-3. DMA0 requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
TIMER0	•	TIMER0_CH0	TIMER0_CH1	TIMER0_CH3 TIMER0_TG TIMER0_CMT	TIMER0_UP	TIMER0_CH2	•
TIMER1	TIMER1_CH2	TIMER1_UP	•	•	TIMER1_CH0	•	TIMER1_CH1 TIMER1_CH3
TIMER2	•	TIMER2_CH2	TIMER2_CH3	•	•	TIMER2_CH0	•
			TIMER2_UP			TIMER2_TG	
TIMER3	TIMER3_CH0	•	•	TIMER3_CH1	TIMER3_CH2	•	TIMER3_UP
ADC0	ADC0	•	•	•	•	•	•
SPI/I2S	•	SPI0_RX	SPI0_TX	SPI1/I2S1_RX	SPI1/I2S1_TX	•	•
USART	•	USART2_TX	USART2_RX	USART0_TX	USART0_RX	USART1_RX	USART1_TX
I2C	•	•	•	I2C1_TX	I2C1_RX	I2C0_TX	I2C0_RX

It should be possible to do this the DMA has enough channels for every event, the timer is not yet marked because I'm not certan at the moment whicht timer and timer signal I will use, I think timer1_up at DMA channel 1.

Timer DMA config:

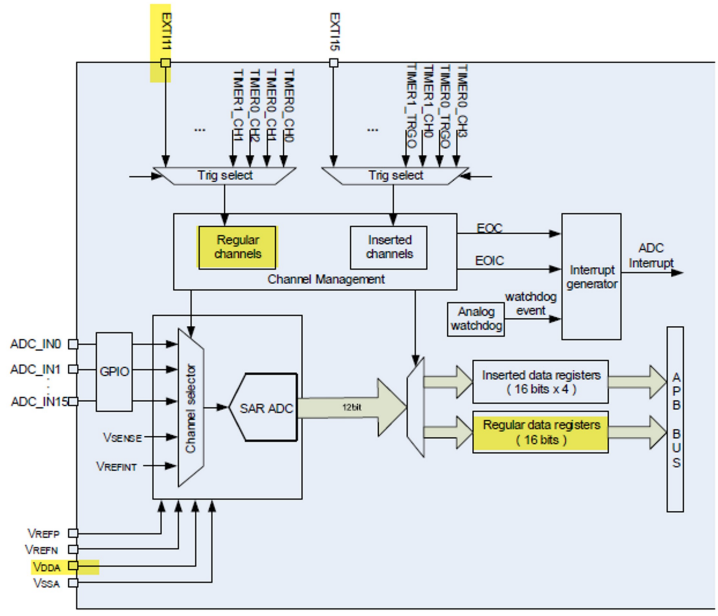
TIMER1_UP -> DMA request on channel 1;

8-12-2021 13:15 It works, see git.

ADC External trigger config

Page 216 user manual

"Single mode converts selected inputs once per trigger."



What is the difference between an injected and regular STM32 ADC channel?
You can configure the ADC to read in a sequence of channels in a loop. Those channels are being converted regularly. In injected mode conversion is triggered by an external event or by software. An injected conversion has higher priority in comparison to a "regular" conversion and thus interrupts the regular conversions.

The different ADC-Modes are explained in application note AN3116.
https://electronics.stackexchange.com/questions/83426/what-is-the-difference-between-an-injected-and-regular-stm32-adc-channel

ADC calibration?
Optional, the sampled signal amplitude is not important. Still a calibration at startup does no harm.

ADC CLK
The ADCCLK clock provided by the clock controller is synchronous with the AHB and APB2 clock. The maximum frequency is 14MHz.

The MCU is running at the maximum freq 108MHz:
108/14=7.7143, so the ADC clock source musthave a prescaler of 8

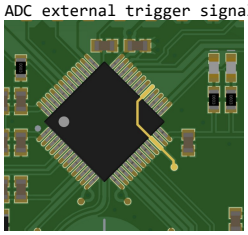
108/8=13.5 MHz

((13.5E6/55)^-1)*1E6=4.0741 uS to take a sample

When the period = 2mS (512T)
A single period =~4uS

The bias is released for 3T and the ADC trigger signal is generated at the second T, so approximatele 2T time is left to sample the signal (when taking 55 points).

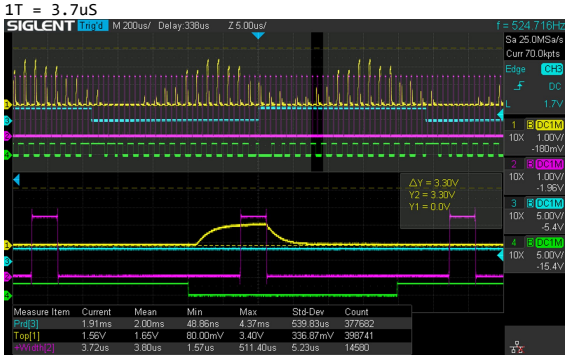
ADC external trigger signal. It is generated by the OCTL so it should be present. There is a test point so you could measure it.



Mapping the ADC trigger to the bias pin so it is easy to monitor without soldering to the test pad
I changed the OCTL generate function to do this while debugging.

Signal sample timing
The ADC is running at 13.5 MHz => T=1/13.5=0.0741 uS per ADC clk cycle

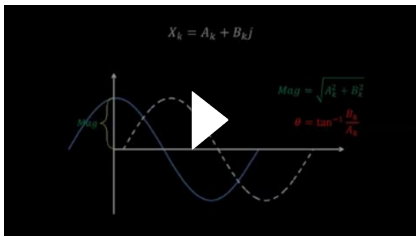
When using prescaler 200 for timer1 sensor timebase.
512T = 1.91mS
1T = 3.7uS



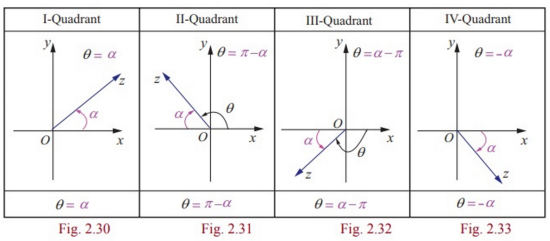
When the ADC trigger is aligned as shown above the maximum sample time is:
3.7/(1/13.5)=49.95

IDE tip: Collapse all function in C file, for easy overview
https://stackoverflow.com/questions/42660670/collapse-all-methods-in-visual-studio-code#:~:text=Windows%3A%20Ctrl%20%2B%20K%20Ctrl%20%2B%20J,%3A%20%E2%8C%98%20%2B%20K%20%E2%8C%98%20%2B%20J

Signal processing
Discrete Fourier Transform - Simple Step by Step

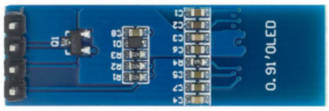


DFT over single frequency bin.



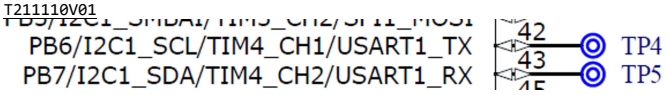
Oled display

128x32 Oled display, SSD1306 driver.



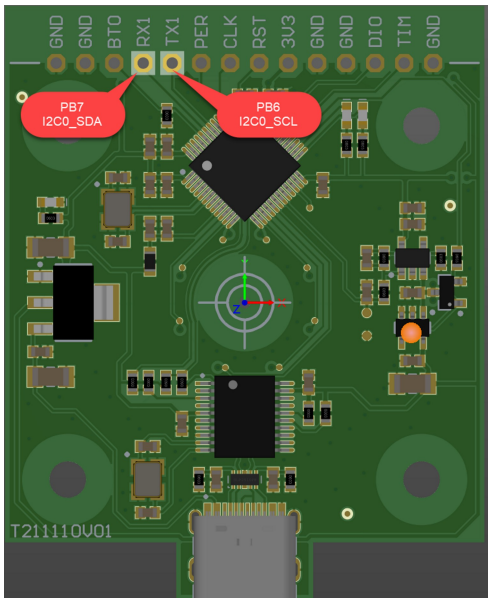
[Aliexpress]
[Waveshare wiki]

T211110V01

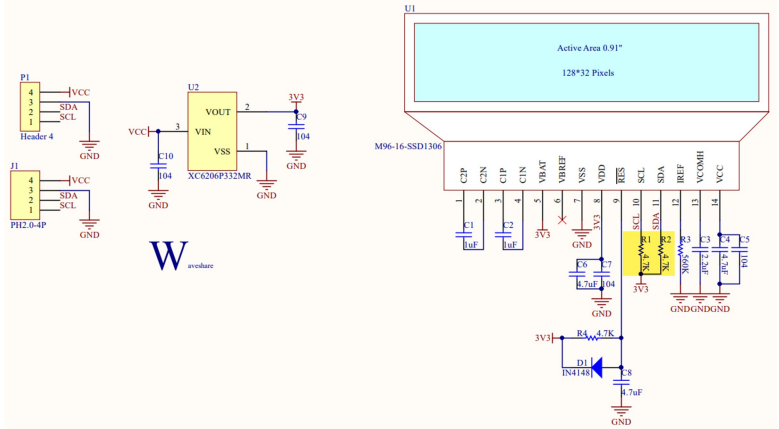


GD32F103xx Datasheet

Pin Name	Pins	Pin Type(1)	I/O Level(2)	Functions description
PB6	42	I/O	5VT	Default: PB6 Alternate: I2C0_SCL, TIMER3_CH0(4) Remap: USART0_TX
PB7	43	I/O	5VT	Default: PB7 Alternate: I2C0_SDA, TIMER3_CH1(4) Remap: USART0_RX

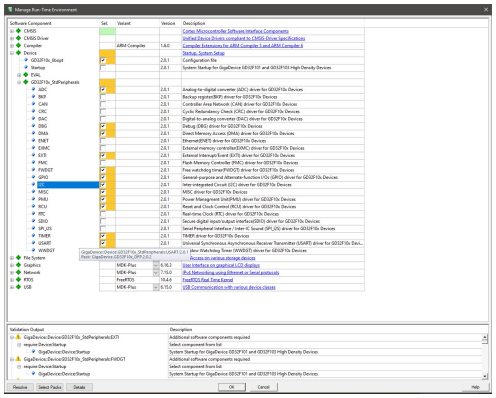


The I2C bus needs Pullups to vcc to function. The display module should have these installed:



12-12-2021 17:41 I have checked using multimeter, and the pullups are present.

Software
Enable i2c driver



I2c scanner to check display address.

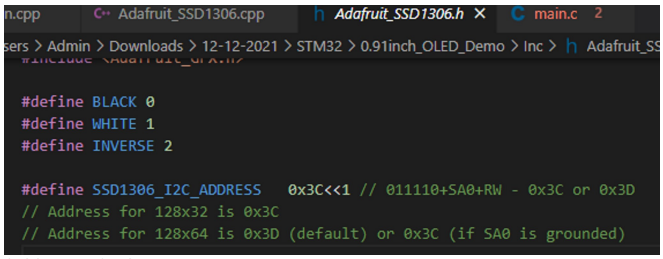
To do this I need to check if an address is acknowleged after it is send.

Table17-3. I2C error flags

I2C Error Name	Description
BERR	Bus error
LOSTARB	Arbitration lost
OUERR	Over-run or under-run when SCL stretch is disabled.
AERR	No acknowledge received

12-12-2021 19:56 : Code for scanning I2c bus is working

Check in code what the Oled display address should be:



0x3C<<1 = 0x78
0x3D<<1 = 0x7A

An I2C scan, to check which adresses are ack
..... 0X78 0X79 .....

0x78 is the oled display.
0x79 Must also be the oled display, because no other devices are used on the bus (check later)

OLED display driver (SSD1306)
After trying to import the adafruit_SSD1306 library in my project, I discovered I need to convert my main.c to main.cpp to use this library.
This was causing problems with my printf UART redirect. And makes microLIB usage not possible (not supported by C++)

I would like to avoid mixing cpp and c in my project so I will look for an other solution.

The following library is written in C, so I will try to add this to my project.
https://github.com/afiskon/stm32-ssd1306