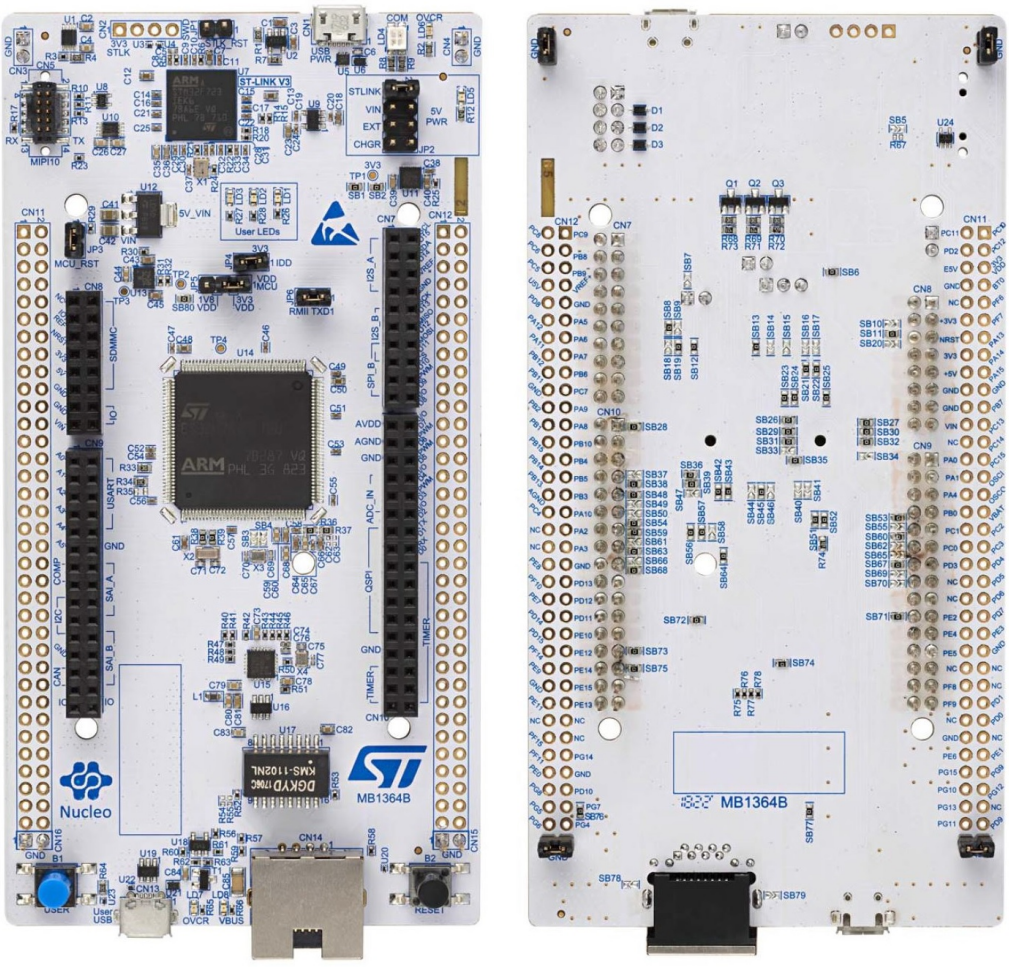
OGS Istituto Nazionale di Oceanografia e di Geofisica Sperimentale

Sezione di geofisica **GEO**

Scheda STM NUCLEO-H743ZI2



Sviluppo software per test scheda GEOPAD

Report OGS 2023/5 Sez. GEO 3

Borgo Grotta Gigante TS 7 febbraio 2023

OGS Istituto Nazionale di Oceanografia e di Geofisica Sperimentale

Sezione di geofisica **GEO**

Scheda STM NUCLEO-H743ZI2

Sviluppo software per test scheda GEOPAD

Autori

Franco Zgauc

Report OGS 2023/5 Sez. GEO 3

Autorizzato
Fausto Ferraccioli

Indice

1. Introduzione

La scheda STM NUCLEO-H743ZI2 è stata scelta come sistema di controllo del modulo di acquisizione GEOPAD di cui il primo prototipo (GEOPAD 1.00) è stato descritto nella relazione interna "OGS 2018-75 GEO 9 GEOP".

Per poter procedere alla progettazione del modulo di acquisizione, ossia la versione finale del modulo GEOPAD, è stato sviluppato un software di test che ha premesso di mettere a punto un protocollo di comunicazione sulla porta ethernet, l'assegnazione delle linee di controllo delle sei porte SPI (SCLK, MISO, MOSI) e l'assegnazione di ulteriori segnali di controllo (DRDY, SYNC, MFLAG, TB, RESET).

In questa relazione si descrivono le parti essenziali del software sviluppato, in particolare l'inizializzazione del microcontrollore, che definisce la configurazione dei collegamenti necessari al controllo del nuovo modulo GEOPAD, il protocollo di comunicazione tramite la porta ethernet e la soluzione di alcune criticità emerse nello sviluppo dovute sia alla complessità di utilizzo che a degli errori presenti in alcune librerie.

2. Caratteristiche della scheda NUCLEO-H743ZI2

La scheda STM NUCLEO-H743ZI2 rappresenta un buon punto di partenza per lo sviluppo di un prototipo. Lo schema a blocchi dell'hardware della scheda (Figura 2.1) evidenzia le caratteristiche principali.

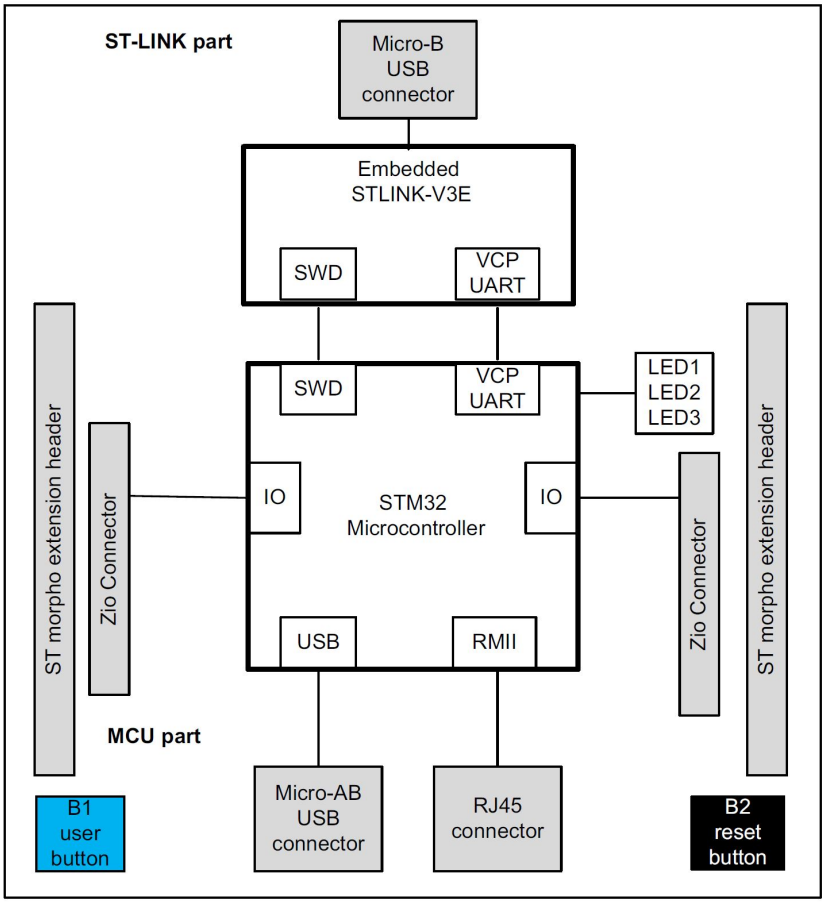


Figura 2.1: Schema a blocchi dell'hardware

Il connettore di espansione *ZIO connector*, che estende il connettore *Arduino Uno V3*, offre un numero di collegamenti al microcontrollore più che sufficienti per la maggior parte dei moduli di espansione. E' inoltre presente un'estensione *ST morpho* a cui sono collegati praticamente tutti i piedini del microcontrollore ma che di default non ha montato il connettore.

La scheda include anche l'*STLINK-V3* debugger/programmer e può quindi facilmente essere programmata e testata da un PC con un semplice collegamento USB che funge anche da presa di alimentazione.

Le caratteristiche più interessanti nell'ambito del progetto GEOPAD sono sicuramente la disponibilità di tutte e sei le porte SPI sul *Zio Connector* e la presenza di una porta ethernet compatibile con lo standard IEEE-802.3-2002.

Per la comunicazione 10M/100M Ethernet è utilizzato un PHY LAN8742A-CZ-TRT ed un connettore RJ45. Per attivare il collegamento bisogna posizionare su ON JP6 e SB72.

La scheda può essere alimentata in vari modi, come già detto l'alimentazione tramite la porta USB è senza dubbio la più semplice da utilizzare durante la fase di debug. Per la versione finale invece risulta più comodo sfruttare una delle connessioni a sorgenti esterne, in particolare la VIN (7V-12V) che permette di collegarsi alla stessa sorgente di alimentazione della scheda GEOPAD (12V).

3. Sistema di sviluppo STM32CubeIDE

STM fornisce gratuitamente un sistema di sviluppo (*STM32CubeIDE*) che semplifica di molto la programmazione ed il test della scheda. Sono disponibili inoltre molte librerie per la gestione del microcontrollore ed esempi di utilizzo.

STM32CubeIDE è una piattaforma di sviluppo in C/C++ che raggruppa in un unico programma la configurazione delle periferiche, la generazione del codice, la compilazione ed il debug del microcontrollore. Inoltre, essendo basata sulla struttura ECLIPSE, dispone di centinaia di plugins per espanderne le funzionalità. La versione utilizzata per questo progetto è la 1.3.1 build 6291_20200406_0752 che integra anche il *Device Configuration Tool* che in versioni precedenti veniva fornito come un programma a se stante.

La creazione di un progetto richiede l'associazione ad una particolare scheda (nel nostro caso la NUCLEO-H743ZI2) in modo da permettere al sistema di creare fin da subito il codice di inizializzazione. Si procede quindi alla configurazione del clock, all'associazione di una particolare funzione a ciascuno dei piedini del microcontrollore, all'aggiunta di eventuali "middleware", ossia di pacchetti software specializzati come ad esempio LwIP (lightweight IP) per la gestione dello stack TCP/IP, ottenendo alla fine un codice di inizializzazione personalizzato.

Lo sviluppo del programma parte da questo codice generato automaticamente in cui sono previste moltissime sezioni per eventuali personalizzazioni contraddistinte come dei blocchi delimitati da un commento di inizio nella forma:

```
/* USER CODE BEGIN .... */
```

ed un commento di fine nella forma:

```
/* USER CODE END ... */
```

Bisogna fare molta attenzione a non inserire del codice al di fuori delle zone assegnate allo USER CODE in quanto queste sono riscritte ogni qual volta si modifica qualche configurazione e vengono quindi rimosse eventuali modifiche inserite manualmente.

A questi sorgenti che si occupano principalmente dell'inizializzazione del microcontrollore si devono aggiungere poi quelli specifici del progetto.

Già in queste prime fasi il codice contiene tutte le istruzioni necessarie all'inizializzazione del microcontrollore ed è quindi possibile, sempre con l'utilizzo del sistema di sviluppo, procedere alla programmazione della scheda ed al successivo debug. Per queste fasi è sufficiente collegare la scheda al PC tramite un cavo USB visto che sulla NUCLEO-H743ZI2 è presente l'*STLINK-V3*.

Una volta creato un progetto il sistema di sviluppo scarica automaticamente da internet l'ultima versione del *firmware package* associato alla scheda utilizzata. Il package, oltre al firmware, contiene molti esempi di utilizzo, librerie specifiche per la scheda e documentazione. Risulta quindi utile poter

accedere a questa cartella la cui posizione è definita in *Window-Preferences-STM32Cube-Firmware Updater-Firmware installation repository*.

Il *firmware package* scaricato per la scheda NUCLEO-H743ZI2, che può anche essere scaricato direttamente dal sito della STM (www.st.com/stm32cubeH7), è il STM32Cube_FW_H7_V1.5.0. Tra le varie risorse disponibili nella directory Drivers\BSP\STM32H7xx_Nucleo (dove BSP sta per Board Specific Package) si trova una libreria molto utile in quanto contiene le API (Application Programming Interface) relative ai componenti hardware della scheda e conviene quindi copiarla all'interno del progetto.

4. Configurazione delle periferiche

4.1 Clock

Il primo passo nello sviluppo del programma è la configurazione del clock che dovendo essere distribuito tra decine di blocchi interni al microcontrollore risulta alquanto complessa. Nel *Device Configuration Tool* è presente una sezione dedicata a questa fase della programmazione che si presenta graficamente come nella figura 4.1. La modifica dei vari parametri porta molto spesso a configurazioni non valide per cui si deve procedere con ripetuti aggiustamenti fino a trovare la versione finale. Le scelte fatte per questo progetto derivano in buona parte dall'analisi di alcuni esempi di utilizzo della porta ethernet e delle porte SPI.

Per poter procedere con la configurazione bisogna assegnare le sorgenti che possono essere sia interne che esterne. Non essendoci particolari necessità di sincronizzazione con altri sistemi sono state utilizzate le sorgenti interne, ossia l'oscillatore a 32,768 kHz per il clock a bassa velocità (LSE clock) e l'oscillatore dell'ST-LINK a 8 MHz per il clock ad alta velocità (HSE clock). A partire dal clock HSE ed utilizzando tre blocchi PLL opportunamente programmati si ottiene un primo insieme di frequenze:

SYSCLK	400 MHz
PLL1Q	400 MHz
PLL2P	100 MHz
PLL2Q	100 MHz
PLL3P	96 MHz
PLL3Q	48 MHz

Da queste passando per vari divisori e multiplexer si arriva alla definizione dei clock per i singoli blocchi. La configurazione finale ottenuta assegna alla CPU un clock a 400 MHz, leggermente inferiore alla frequenza massima del microcontrollore (480MHz), mentre all'*Internal Clock* utilizzato come base per tutte le periferiche 200 MHz. A partire da quest'ultimo clock mediante alcuni divisori (D1PPRE, D2PPRE1, D2PPRE2 e D3PPRE) si ottengono i clock di riferimento per tutte le periferiche (APB3, APB1 APB2 e APB4), utilizzando l'opzione /2 per tutti i divisori si ottiene un'unica frequenza di riferimento pari a 100MHz.

Da ultimo si passa ad assegnare il clock alle varie periferiche facendo attenzione però che fintanto che non si attiva una particolare periferica la sezione corrispondente rimane disattivata e quindi non configurabile. Una volta attivate le sei porte SPI è possibile associare a ciascuna un clock che, per semplificare la programmazione, è posto per tutte a 100MHz; in particolare alle porte SPI1, SPI2

e SPI3 si associa il clock PLL2P (prima linea del PLL2), alle porte SPI4 e SPI5 il clock PCLK2 (collegato alla linea APB2) e alla porta SPI6 il clock PCLK4 (collegato alla linea APB4).

Nella figura 4.1 è riportata la pagina di configurazione dopo aver apportato le modifiche appena descritte.

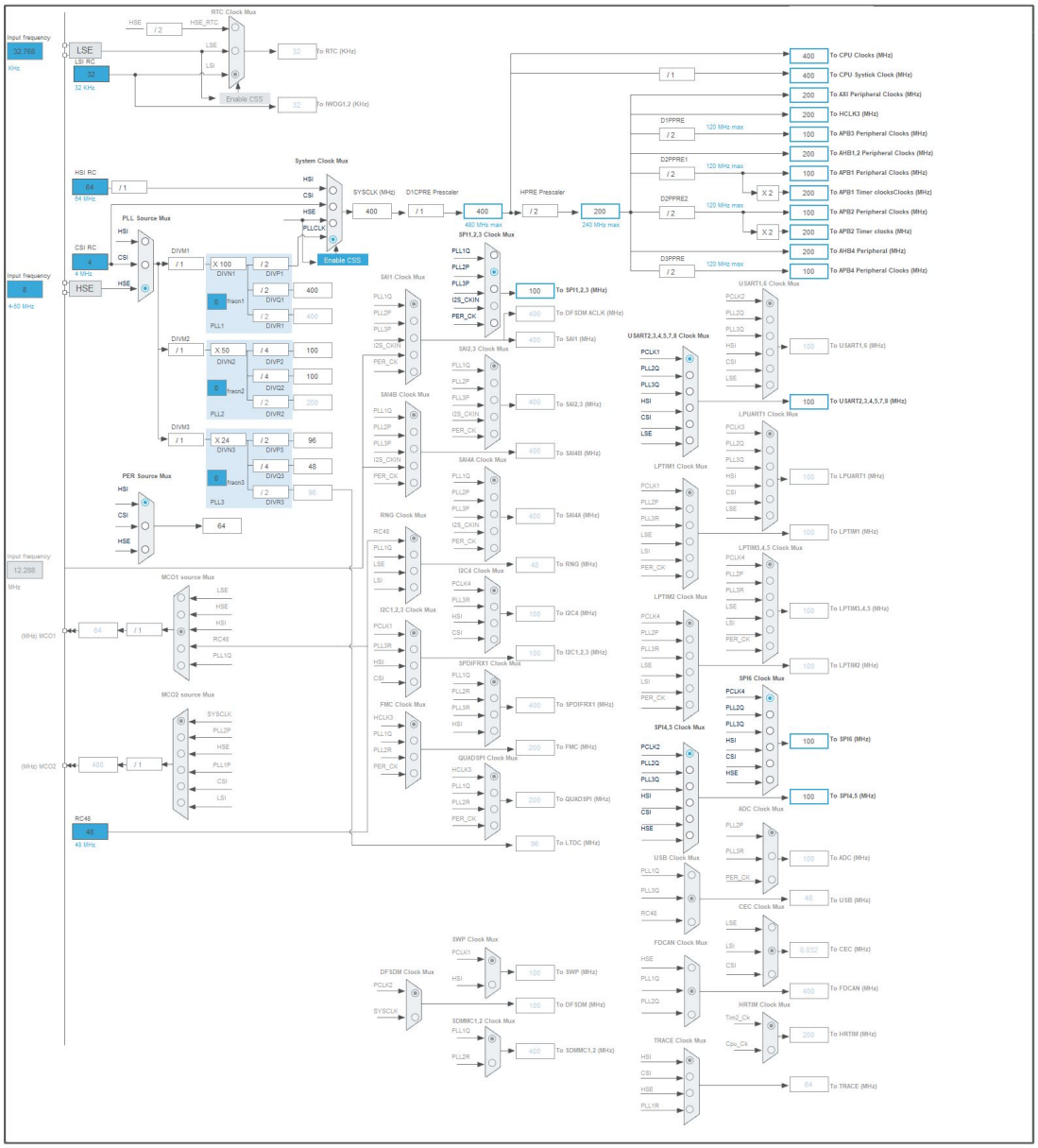


Figura 4.1: Configurazione del clock

4.2 Microcontrollore e memoria

Nella sezione *System Core-CORTEX_M7-Parameter Settings* bisogna effettuare diverse configurazioni necessarie per il corretto funzionamento del middleware LwIP (capitolo 5.0.1). In primo luogo bisogna attivare le due cache della CPU altimenti LwIP risulta addirittura non installabile. Per le zone di memoria la configurazione è legata direttamente alle caratteristiche di funzionamento della porta ETH (capitolo 4.5.1). L'*ETH DMA* non può accedere alla memoria DTCM RAM e quindi tutti i dati gestiti (*TX Descriptor*, *RX Descriptor* e buffer di ricezione e trasmissione) devono essere scritti nella memoria SRAM. Bisogna poi considerare che il microcontrollore (Cortex-M7) può accedere alla memoria in modalità *out-of-order* e quindi ci potrebbero essere degli scambi d'ordine nelle sequenze di scrittura nella SRAM e nei registri Ethernet, quindi i *TX e RX DMA descriptors* devono trovarsi in una memoria configurata come *Device* o *Strongly-ordered*. Per tener conto di questi problemi STM suggerisce di utilizzare i seguenti valori:

RX buffer used by the Ethernet driver	16 kb at 0x30040000 configured as non-cacheable, MPU region 0 (required for overlapping)
RX buffers allocated by LwIP	16 kb at 0x30044000 configured as write-through, MPU region 1
RX e TX DMA descriptors	256 bytes at 0x30040000 configured as Sharde Device, MPU region 2 (required for overlapping)

che porta alla seguente configurazione:

Cortex Interface Settings	
CPU ICache	Enabled
CPU DCache	Enabled
Cortex Memory Protection Unit Control Settings	
MPU Control Mode	Background Region Privileged accesses only + MPU Disabled during hard fault, NMI and FAULTMASK handlers
Cortex Memory Protection Unit Region 0 Settings	
MPU Region	Enabled
MPU Region Base Address	0x30040000
MPU Region Size	256B
MPU SubRegion Disable	0x0
MPU TEX field level	level 0
MPU Access Permission	ALL ACCESS PERMITTED
MPU Instruction Access	ENABLE
MPU Shareability Permission	DISABLE
MPU Cacheable Permission	DISABLE
MPU Bufferable Permission	ENABLE
Cortex Memory Protection Unit Region 1 Settings	
MPU Region	Enabled
MPU Region Base Address	0x30044000
MPU Region Size	16KB
MPU SubRegion Disable	0x0
MPU TEX field level	level 1
MPU Access Permission	ALL ACCESS PERMITTED

MPU Instruction Access	ENABLE
MPU Shareability Permission	ENABLE
MPU Cacheable Permission	DISABLE
MPU Bufferable Permission	DISABLE

Per posizionare queste zone di memoria nel blocco di memoria D2, accessibile all'*ETH DMA*, bisogna aggiungere al linker script (file STM32H743ZITX_FLASH.id) le seguenti righe:

```
.lwip\_sec (NOLOAD) :
{
  . = ABSOLUTE(0x30040000);
  *(.RxDecripSection)

  . = ABSOLUTE(0x30040060);
  *(.TxDecripSection)

  . = ABSOLUTE(0x30040200);
  *(.RxArraySection)
} >RAM\_D2 AT> FLASH
```

dove il nome e l'indirizzo di ciascuno dei tre blocchi di memoria deve ovviamente coincidere con quello definito nel sorgente ethernetif.c.

4.3 Interrupt

Nella sezione *System Core-NVIC* (Nested Vectored Interrupt Controller), oltre agli interrupt di sistema che sono preattivati, sono state attivate le seguenti linee:

EXTI line0 interrupt
EXTI line[9:5] interrupts
EXTI line[15:10] interrupts
TIM3 global interrupts
Ethernet global interrupt

Le attivazioni *EXTI line0*, *EXTI line[9:5]* e *EXTI line[15:10]* permettono di associare un interrupt a dei pin di input, ogni singola linea corrisponde al pin dello stesso numero in una delle porte del microcontrollore. Bisogna quindi fare attenzione a non collegare segnali di interrupt a pin con lo stesso numero su porte diverse. Le porte utilizzate, come indicato nel capitolo 4.6, sono PE0, PG6, PB11 e PC13 che corrispondono quindi alle linee di interrupt 0, 6, 11 e 13.

La linea *TIM3 global interrupt* è attivata tramite la configurazione del timer TIM3, così come descritto in dettaglio nel capitolo 4.4. Similmente *Ethernet global interrupt* dipende dalla configurazione della porta ETH, capitolo 4.5.1

4.4 Timer

Di default il sistema di sviluppo propone come sorgente per la gestione delle periferiche il *SysTick* di sistema, questa scelta però porta spesso a malfunzionamenti in particolare quando si utilizzano molte

periferiche o si installa il sistema operativo RTOS. Per questo motivo nella sezione *System Core-SYS* è stato selezionato come *Timebase Source* il timer TIM1.

Nel programma vengono poi utilizzati altri due timer, il TIM2 per generare un contatore con precisione al microsecondo ed il TIM3 per generare un interrupt ogni millisecondo. Il primo viene utilizzato nella comunicazione con gli AD converter, il secondo per far lampeggiare un led.

Nella configurazione di TIM2 si utilizza come sorgente l'*Internal Clock* a 200MHz. Per ottenere un contatore che si incrementa ogni microsecondo bisogna dividere la frequenza in ingresso per 200 e quindi si setta il *Prescaler* a 199 (200-1). L'interrupt è disabilitato in quanto il timer è utilizzato come contatore.

Mode	
Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Parameter Setting	
Counter Settings	
Prescaler (PSC - 16 bits value)	200-1
Counter Mode	Up
Counter Period (AutoRelaod Register - 32 bits value)	0xffffffff-1
Internal Clock Division (CKD)	No Division
auto-relaod preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO	Reset (UG bit from TIMx_EGR)
NVIC Settings	
TIM2 global interrupt	Disabled

Anche per TIM3 si utilizza come sorgente l'*Internal Clock* a 200MHz. In questo caso però si vuol generare un interrupt ogni millisecondo, quindi l'interrupt è abilitato, il *Prescaler* è posto a 19999 (20000-1) per ottenere una frequenza base di 10 KHz e il *Counter Period* è posto a 9 (10-1) per generare l'interrupt ogni 10 passi del clock.

Mode	
Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Counter Settings	
Prescaler (PSC - 16 bits value)	19999
Counter Mode	Up
Counter Period (AutoRelaod Register - 16 bits value)	9
Internal Clock Division (CKD)	No Division
auto-relaod preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO	Enable (CNT_EN)
NVIC Settings	
TIM3 global interrupt	Enabled

4.5 Connettività

Nella sezione connettività sono state attivate le seguenti periferiche: ETH, SPI1, SPI2, SPI3, SPI4, SPI5, SPI6 e USART3. La porta ethernet e le porte SPI sono ovviamente necessarie per il funzionamento del sistema mentre la porta USART3 serve per inviare delle informazioni di debug.

4.5.1 Porta ETH

Come accennato nel capitolo 2 la porta ethernet è già presente sulla scheda e quindi quasi tutti i parametri sono predefiniti. La modalità di collegamento tra la CPU ed il chip PHY LAN8742A-CZ-TR è la RMII (Reduced Media Independent Interface) ed i pin assegnati sono i seguenti:

Pin Name	**Signal on Pin**	**Maximum output speed**
PA1	ETH_REF_CLK	Very High
PA2	ETH_MDIO	Very High
PA7	ETH_CRS_DV	Very High
PB13	ETH_TXD1	Very High
PC1	ETH_MDC	Very High
PC4	ETH_RXD0	Very High
PC5	ETH_RXD1	Very High
PG11	ETH_TX_EN	Very High
PG13	ETH_TXD0	Very High

dove a tutti i pin è stato assegnato il valore Very High nel parametro *Maximum output speed* al posto del valore Low di default. Per quanto riguarda i parametri generali (MAC Address e mappatura nella memoria) si sono utilizzati i valori di default.

General Ethernet Configuration

Ethernet MAC Address	00:80:E1:00:00:00
Tx Descriptor Lenght	4
First Tx Descriptor Address	0x30040060
Rx Descriptor Lenght	4
First Rx Descriptor Address	0x30040000
Rx Buffers Address	0x30040200
Rx Buffers Length	1536

4.5.2 Porte SPI

Le porte SPI servono alla comunicazione con gli AD converter presenti nella scheda GEOPAD e quindi devono essere configurate sostanzialmente allo stesso modo. Il *Mode* utilizzato è *Full-Duplex Master* e nella sezione *Parameter Settings* i valori risultano uguali per tutte e sei le porte SPI:

Basic Parameters

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

Clock Parameters	
Prescale (for Baud Rate)	64
Baud Rate	1.5625 MBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled
NSSP Mode	Disabled
NSS Signal Type	Software
Fifo Threshold	Fifo Threshold 01 Data
Tx Crc Initialization Pattern	All Zero Pattern
Rx Crc Initialization Pattern	All Zero Pattern
Nss Polarity	Nss Polarity Low
Master Ss Idleness	00 Cycle
Master Inter Data Idleness	00 Cycle
Master Receiver auto Susp	Disable
Master Keep Io State	Master Keep Io State enable
IO Swap	Disabled

Il *Data Size* va posto a 8 (il valore di default è 4) in quanto il protocollo di comunicazione dell'ADS1282 prevede il byte come unità base nello scambio di informazioni. Sempre facendo riferimento alle specifiche dell'ADS1282 la velocità massima di trasmissione sulla porta SPI è pari alla metà del clock dell'AD converter (4,096MHz) e quindi 2,048 Mbits/s. Si è visto nel capitolo 4.1 che a tutte le porte SPI è stato assegnato un clock di 100 MHz, quindi per rimanere all'interno delle specifiche bisogna impostare il *Prescale* a 64 il che porta ad un *Baud Rate* di 1,5625 Mbits/s.

Per quanto riguarda la polarità (CPOL) e la fase (CPHA) i valori selezionati dipendono dal fatto che l'AD converter campiona il dato in ingresso sul fronte di salita del clock e fornisce il dato in uscita sul fronte di discesa. Il *NSSP Mode* (Negative Slave Select Pulse) deve essere disabilitato visto che l'ADS1282 non dispone di un pin di CS. Da ultimo il *Master Keep Io State* deve essere abilitato in modo che i segnali in uscita siano sempre sotto il controllo del master, se disabilitato i pin non sono pilotati durante le fasi di inattività della porta SPI (tri-state).

Anche nella sezione *GPIO Settings* ci sono alcuni valori comuni a tutte le porte e sono riportati nella seguente tabella:

Pin Context Assignement	n/a
GPIO output level	n/a
GPIO mode	Alternate Function Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed.	Low
Fast Mode	n/a

sono invece ovviamente diversi quelli relativi all'assegnazione dei pin. La soluzione adottata è sostanzialmente quella proposta di default dal sistema di sviluppo, l'unico valore modificato è quello relativo al segnale SPI3 MOSI visto che con il valore di default (PB2) sono emersi diversi malfunzionamenti durante il debug.

Pin Name	Signal on Pin
PA5	SPI1_SCK
PA6	SPI1_MISO
PD7	SPI1_MOSI
PB10	SPI2_SCK
PC2_C	SPI2_MISO
PC3_C	SPI2_MOSI
PC10	SPI3_SCK
PC11	SPI3_MISO
PC12	SPI3_MOSI
PE2	SPI4_SCK
PE5	SPI4_MISO
PE6	SPI4_MOSI
PF7	SPI5_SCK
PF8	SPI5_MISO
PF9	SPI5_MOSI
PB3	SPI6_SCK
PG12	SPI6_MISO
PG14	SPI6_MOSI

4.5.3 Porta USART3

Questa porta, utilizzata principalmente per il debug, non richiede configurazioni particolari ed i parametri sostanzialmente sono quelli di default.

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples
Single Sample	Disable
ClockPrescaler	clock/1
Fifo Mode	Disable
Txfifo Threshold	1 eighth full configuration
Rxfifo Threshold	1 eighth full configuration
Advanced Features	
Auto Baudrate	Disable
TX Pin Active Level Inversion	Disable
RX Pin Active Level Inversion	Disable
Data Inversion	Disable
TX and TX Pins Swapping	Disable
Overrun	Enable
DMA on RX Error	Enable
MSB First	Disable

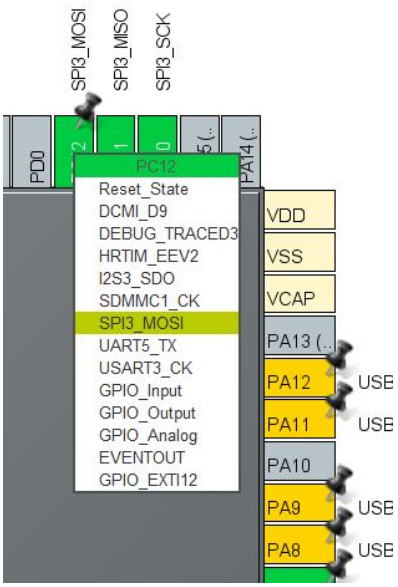


Figura 4.2: Tendina di selezione per l'assegnazione di un pin

4.6 GPIO

Una parte dei pin di connessione è già stata definita con la configurazione delle porte ETH e SPI, rimangono però diversi altri segnali della scheda GEOPAD da associare a dei pin del microcontrollore. Questa operazione può essere eseguita graficamente nella sezione *Pinout view* del *Device Configuration Tool*. Cliccando su un pin si apre una finestra a tendina con un elenco di tutte le possibili configurazioni di quel pin, così come in figura 4.2. Una volta effettuata l'assegnazione il pin compare nell'elenco *System Core-GPIO* in cui è possibile modificare tutti i parametri associati, in particolare si può associare una *User Label* che potrà poi essere utilizzata come nome del pin nel programma.

I segnali da associare sono MFLAG1..6, TB1, TB2, SYNC, DRDY e RESET. Allo stesso modo si associano anche i tre led presenti sulla scheda ed il pulsante blu che nel programma viene utilizzato come time-break.

Pin Name	**User Label**	**GPIO output**	**GPIO mode**
PB0	LEDGREEN	Low	Output Push Pull
PB14	LEDRED	Low	Output Push Pull
PE1	LEDYELLOW	Low	Output Push Pull
PC13	BLUBUTT	n/a	External Interrupt Mode with Rising edge trigger detection
PC8	ADRESET	Low	Output Push Pull
PG6	DRDY1	n/a	External Interrupt Mode with Falling edge trigger detection
PG0	SYNC	Low	Output Push Pull
PB9	MFLAG1	n/a	Input mode

PB8	MFLAG2	n/a	Input mode
PB5	MFLAG3	n/a	Input mode
PB15	MFLAG4	n/a	Input mode
PC9	MFLAG5	n/a	Input mode
PD2	MFLAG6	n/a	Input mode
PE0	TB1	n/a	External Interrupt Mode with Rising edge trigger detection
PB11	TB2	n/a	External Interrupt Mode with Rising edge trigger detection

Nella *Pinout view* (figura 4.3) appaiono evidenziate in un'unica immagine tutte le assegnazioni fatte.

Figura 4.3: Assegnazione dei pin

5. Middleware

Per semplificare al massimo il progetto l'unico *Middleware* utilizzato è il LwIP (lightweight IP) necessario per la gestione dello stack TCP/IP. Per rendere più flessibile il programma si potrebbe includere anche il FreeRTOS, un sistema operativo real-time pensato per i microcontrollori, ma ciò renderebbe inutilmente più complesso il software che al momento serve solo per definire i collegamenti tra le due schede e per verificarne il corretto funzionamento.

5.0.1 LwIP

Di default LwIP è configurato in modalità DHCP (Dynamic Host Configuration Protocol) ossia con l'indirizzo IP assegnato da un server, in questo modo però risulta più complesso identificare la scheda nella rete in particolare se coesistono più schede, si è utilizzato quindi un IP statico. Questa configurazione purtroppo non è gestita correttamente dal *Device Configuration Tool* e richiede degli interventi manuali sul sorgente così come descritto dettagliatamente nel capitolo 6.

Per quanto riguarda il *transport layer* sono stati attivati entrambi i protocolli UDP e TCP in quanto a seconda delle modalità di utilizzo potrebbe risultare utile sfruttare le caratteristiche dell'uno o dell'altro. In questa versione comunque si è utilizzato soltanto il protocollo TCP che è un po' più pesante ma garantisce l'ordine con cui arrivano i pacchetti di dati. Il driver PHY per la gestione del collegamento con il chip di controllo della porta ethernet è il LAN8742 che deve essere settato anche nella sezione *Platform Settings*. Sempre nella sezione *General Settings*, riportata qui di seguito, è stato attivato il modulo ICMP in modo che la scheda risponda al comando *ping*.

LwIP Version	
LwIP Version (CubeMX specific)	2.0.3
IP Address Settings	
IP_ADDRESS (IP Address)	192.168.001.011
NETMASK_ADDRESS (Netmask Address)	255.255.255.000
GATEWAY_ADDRESS (Gateway Address)	192.168.001.254
RTOS Dependency	
WITH_RTOS (Use FREERTOS)	Disabled
Platform Setting	
PHY Driver	Choose/LAN8742
Protocols Options	
LWIP_ICMP (ICMP Module Activation)	Enabled
LWIP_IGMP (IGMP Module)	Disabled
LWIP_DNS (DNS Module)	Disabled

LWIP_UDP (UDP Module)	Enabled
MEMP_NUM_UDP_PCB (Number of UDP Connections)	4
LWIP_TCP (TCP Module)	Enabled
MEMP_NUM_TCP_PCB (Number of TCP Connections)	5

Oltre a queste configurazioni di base bisogna apportare dei cambiamenti anche nella sezione *Key Options*. Una volta attivata l'opzione *Show advanced parameters* il sistema presenta circa un centinaio di parametri e quasi tutti i valori di default risultano corretti per un buon funzionamento del sistema. Di seguito vengono riportate soltanto le sezioni più rilevanti.

Infrastructure - OS Awarness Option	
NO_SYS (OS Awarness)	OS Not Used
Infrastructure - Timers Options	
LWIP_TIMERS (Use Support For sys_timeout)	Enabled
LWIP_TIMERS_CUSTOM (Own Timer Implementation)	Disabled
Infrastructure - Heap and Memory Pools Options	
MEM_LIBC_MALLOC (User Memory Library)	Disabled
MEMP_MEM_MALLOC (User Memory Pool Functions)	Disabled
MEM_ALIGNMENT (Memory Byte Alignment of CPU)	4 Byte(s)
MEM_SIZE (Heap Memory Size)	8192 Byte(s)
MEM_OVERFLOW_CHECK (Memory Pool Overflow Protection)	0
MEMP_SANITY_CHECK (Memory Pool Sanity Check)	Disabled
MEM_USE_POOLS (Use an Alternative to malloc Function)	Disabled
MEM_USE_POOLS_TRY_BIGGER_POOL (Try Next Bigger Pool if Empty...	Disabled
MEM_USE_CUSTOM_POOLS (Define Additional Pools)	Disabled
LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT (Allow Memory Free...	Disabled
LWIP_RAM_HEAP_POINTER (RAM Heap Pointer)	0x30044000
Pbuf Options	
LWIP_SUPPORT_CUSTOM_PBUF (Custom Pbuf)	Enabled
PBUF_POOL_SIZE (Number of Buffers in the Pbuf Pool)	16
PBUF_LINK_HLEN (Number of Bytes for Link Level Header)	14 Byte(s)
PBUF_LINK_ENCAPSULATION_HLEN (Number of Bytes for Link...	0 Byte(s)
PBUF_POOL_BUFSIZE (Size of each pbuf in the pbuf pool)	512
IPv4 - ICMP Options	
ICMP_TTL (Time-To-Live Used by ICMP Packets)	255 Node(s)
LWIP_BROADCAST_PING (Respond to Broadcast Pings)	Enabled
LWIP_MULTICAST_PING (Respond to Multicast Pings)	Enabled
Network Interfaces Options	
LWIP_NETIF_HOSTNAME (NETIF Hostname)	Disabled
LWIP_NETIF_API (NETIF API)	Disabled
LWIP_NETIF_STATUS_CALLBACK (Callback Function on Interface Status...	Disabled
LWIP_NETIF_LINK_CALLBACK (Callback Function on Interface Link Changes)	Disabled
LWIP_NETIF_REMOVE_CALLBACK (Callback Function on Interface Removal)	Disabled
LWIP_NETIF_HWADDRHINT (Cache Link-layer-address Hints)	Disabled
LWIP_NETIF_TX_SINGLE_PBUF (Put Data into One Single Pbuf)	Disabled
LWIP_NUM_NETIF_CLIENT_DATA (Cache Link-layer-address Hints)	0

Nella sezione *Heap and Memory Pools* è stata aumentata le dimensione della *heap memory* a 8192 byte in quanto il valore di default (1600) in alcune situazioni risulta insufficiente, mentre nella sezione *Pbuf Options* è stato ridotto a 512 il valore di PBUF_POOL_BUFSIZE (default 592) in quanto valore consigliato nell'esempio di utilizzo fornito da STM. Riguardo al modulo ICMP per attivare la risposta al comando ping si devono abilitare i parametri LWIP_BROADCAST_PING e LWIP_MULTICAST_PING.

Può inoltre risultare utile l'attivazione del parametro LWIP_NETIF_LINK_CALLBACK per far rilevare al sistema l'inserimento o la rimozione di un cavo nella presa RJ45.

Il pacchetto LwIP è molto complesso e piccoli errori di configurazione portano spesso a dei malfunzionamenti del collegamento ethernet, risulta perciò molto utile attivare almeno in parte le segnalazioni di debug previste. Per abilitare il debug di LwIP è necessario aggiungere nel file lwipopts.h la riga:

```
#define LWIP_DEBUG 1
```

Per attivare o disattivare delle particolari segnalazioni di debug si può poi utilizzare l'apposita pagina dell'LwIP all'interno del *Device Configuration Tool*. La visualizzazione di queste segnalazioni può essere effettuata direttamente sul PC a cui la scheda è collegata tramite porta USB, a tal scopo si deve definire una porta seriale in uscita (vedi capitolo 4.5.3) collegata all'*STLINK-V3* ed inviare a questa lo standard output. La ridirezione dell'output si ottiene inserendo nel file main.c le seguenti righe:

```
#ifdef __GNUC__
    #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
    #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */

PUTCHAR_PROTOTYPE
{
    HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
```

a questo punto bisogna individuare sul PC la porta seriale virtuale associata alla scheda STM che è presente tra le porte COM e LPT con il nome STMicroelectronics STLink Virtual COM Port e quindi attivare un terminale collegato alla porta (p.es. putty).

6. Bachi nel codice autogenerato

Purtroppo il codice generato dal *Device Configuration Tool* non è privo di bachi. Due in particolare, se non corretti, portano ad un malfunzionamento della scheda. La correzione deve essere rifatta ogni qual volta si cambia qualche parametro in quanto il *Device Configuration Tool* ad ogni modifica rigenera tutto il codice eliminando quindi tutte le modifiche apportate manualmente.

Il primo errore si trova nel file *ethernetif.c* (sorgente A.7). Nella funzione *low_level_output* (riga 353) è necessario invalidare la cache prima di chiamare la funzione *HAL_ETH_Transmit* altrimenti la porta ethernet funziona in ricezione ma non in trasmissione.

In pratica bisogna inserire il seguente comando alla riga 387:

```
/* Clean and Invalidate data cache */
SCB_CleanInvalidateDCache();
```

Questo problema è abbastanza discusso su vari forum ed è stato segnalato da tempo ad STM che però al momento non ha fornito una versione funzionante del software. La soluzione qui adottata non risolve del tutto i problemi ed infatti alcuni utenti hanno riscritto parte del driver ed abbandonato quello STM, ma per le finalità dell'attuale versione del programma la correzione risulta sufficiente.

Il secondo errore si trova nel file *lwip.c* (sorgente A.4). Se nella configurazione del modulo middleware LWIP si assegna un indirizzo IP statico nel sorgente è presente il commento iniziale (/* IP addresses initialization */) ma mancano tutte le istruzioni di assegnazione, non correggendo l'errore il programma non inizializza le variabili e generalmente si ci ritrova con tutti i valori a zero.

Bisogna quindi inserire all'inizio della funzione *MX_LWIP_Init* (riga 58) le seguenti righe di programma:

```
IP_ADDRESS[0] = 192;
IP_ADDRESS[1] = 168;
IP_ADDRESS[2] = 1;
IP_ADDRESS[3] = 11;
NETMASK_ADDRESS[0] = 255;
NETMASK_ADDRESS[1] = 255;
NETMASK_ADDRESS[2] = 255;
NETMASK_ADDRESS[3] = 0;
GATEWAY_ADDRESS[0] = 192;
GATEWAY_ADDRESS[1] = 168;
GATEWAY_ADDRESS[2] = 1;
GATEWAY_ADDRESS[3] = 254;
```

7. Software

In questo capitolo vengono descritte le parti più rilevanti del software sviluppato. Molti dei file sono generati direttamente dal *Device Configuration Tool* e contengono principalmente il codice necessario alla gestione del livello più basso del software indicato con l'acronimo HAL (Hardware Abstraction Layer).

A parte qualche modifica nei file autogenerati la maggior parte del programma sviluppato si trova nei file SPI.c e tcp_server.c che gestiscono, rispettivamente, il protocollo di comunicazione sulla porta SPI e quello sulla porta ethernet.

Essendo un numero elevato di file qui di seguito è riportata una breve descrizione del contenuto di ciascuno.

Nome file	Descrizione
main.c	Il programma principale. E' generato dal *Device Configuration Tool* ma contiene anche diverse parti aggiuntive
main.h	Header di main.c
lwip.c	Codice di inizializzazione di LwIP, generato dal *Device Configuration Tool* è stato modificato solo per correggere un errore come indicato nel capitolo 6
lwip.h	Header di lwip.c
lwipopts.h	Permette di modificare la configurazione di default di LwIP che si trova nel file di sistema opt.h
lwippools.h	Inizializzazione di LwIP
ethernetif.c	Gestione dell'interfacciamento tra LwIP ed il driver ETH
ethernetif.h	Header di ethernetif.c
stm32h7xx_hal_msp.c	Gestione MSP (MCU Support Package). Inizializza le periferiche (SPI) e le risorse a basso livello del microcontrollore (clock, DAC, TIM, UART)
stm32h7xx_hal_timebase_tim.c	Gestione della base dei tempi di HAL (Hardware Abstraction Layer). Utilizza il timer TIM1 come sorgente per generare un interrupt ogni millisecondo.
stm32h7xx_hal_conf.h	Header per i moduli di configurazione di HAL (Hardware Abstraction Layer)
stm32h7xx_it.c	Gestione degli interrupt
stm32h7xx_it.h	Header file di stm32h7xx_it.c
syscalls.c	Un insieme minimo di funzioni di input/output
sysmem.c	Un insieme minimo di funzioni di accesso alla memoria interna del microcontrollore
system_stm32h7xx.c	Contiene le funzioni di configurazione del clock e di inizializzazione di alcune variabili di sistema. Sono funzioni eseguite subito dopo il reset e prima dell'avvio del programma principale
SPI.c	Gestione del protocollo di comunicazione sulle porte SPI
SPI.h	Header di SPI.c
tcp_server.c	Gestione del protocollo di comunicazione sulla porta ethernet
tcp_server.h	Header file di tcp_server.c
STM32H743ZITX_FLASH.ld	Linker script. Definizione di tutti i banchi di memoria e dimensioni di heap e stack

7.1 Programma principale

Il programma principale consiste essenzialmente di una lunga serie di chiamate a funzioni di inizializzazione di tutte le sezioni del microcontrollore e termina con un loop infinito che si limita a monitorare il flusso dati sulla porta ethernet. Alle funzioni autogenerate sono state aggiunte le inizializzazioni del gestore dei protocolli TCP e UDP, un timer con precisione del microsecondo e la gestione di un led.

7.2 Protocollo ethernet

Il programma prevede che la scheda funzioni sempre come client, quindi tutto il flusso dati dipende dai comandi impartiti dal server via ethernet. Il protocollo di comunicazione consiste in una decina di semplici comandi sufficienti però a verificare il corretto funzionamento della scheda. Tutte le routine sono contenute nel file tcp_server.c ottenuto modificando un semplice esempio di utilizzo di LwIP (TCP echo server).

Comando	**Descrizione**
'r'	Lettura del contenuto dei registri di un AD converter
'w'	Scrittura di un registro di un AD converter
'b'	Richiesta invio dei dati di un buffer
'i'	Richiesta informazioni sul firmware
'c'	Scrittura dei parametri di calibrazione dell'offset
'g'	Scrittura dei parametri di calibrazione del gain
'p'	Attende il time-break per acquisire un record
'n'	Forza l'inizio dell'acquisizione di un record
'f'	Acquisizione continua
's'	Definizione del numero di campioni contenuti in un record
'x'	Reset dell'AD converter

7.2.1 Comando 'r' (lettura registri interni)

Questo comando legge tutti i registri interni dell'ADS1282 indicato.

'r'	ADnum

Il comando prevede un parametro (ADnum) che indica il numero dell'AD converter (da 1 a 6) a cui richiedere il valore dei registri.

La risposta è una stringa di 13 byte con il seguente formato:

'R'	ID	CONFIG0	CONFIG1	HPF0	HPF1	OFC0	OFC1	OFC2	FSC0	FSC1	FSC2	NL

al carattere 'R' iniziale seguono i valori di tutti i parametri ed il carattere NL (new line, valore 10 nella tabella ASCII) come delimitatore finale.

7.2.2 Comando 'b' (invio dati)

Questo comando richiede l'invio dei dati dei buffer di acquisizione.

'b'

A ciascuno dei 6 AD converter è associato un buffer di acquisizione gestito come una coda circolare di 4 elementi. In ogni elemento vengono memorizzati consecutivamente 256 campioni di 4 byte ciascuno, quindi la dimensione di ogni elemento è pari a 1024 byte. All'arrivo di una nuova richiesta di invio dati il programma confronta la posizione del puntatore di svuotamento, quello cioè che punta all'elemento da inviare, con il puntatore di riempimento, che punta invece all'elemento in cui inserire il prossimo campione. Se i due puntatori sono diversi significa che l'elemento richiesto è pronto per l'invio e si procede quindi con l'invio del dato, se sono uguali invece significa che non è arrivato ancora un numero sufficiente di campioni e quindi si risponde negativamente. L'acquisizione avviene in maniera sincrona su tutti i canali per cui è sufficiente controllare lo stato di uno solo.

Ovviamente se la richiesta dei dati avviene troppo lentamente si corre il rischio che un elemento venga sovrascritto prima di essere scaricato. Nella versione attuale del software questo evento non genera nessun errore, ad ogni lettura però viene fornito un indice sequenziale dell'elemento letto che permette al server di controllare la correttezza della sequenza.

Se i due puntatori sono diversi la risposta è una stringa di 6150 byte con il seguente formato:

'D'	Counter	AD1data	AD2data	AD3data	AD4data	AD5data	AD6data	NL

al carattere 'D' iniziale segue il valore del contatore sequenziale (Counter di 4 byte), quindi i sei blocchi dati da 1024 byte ciascuno ed il NL di chiusura.

Nel caso in cui i due puntatori della coda coincidano non è possibile inviare alcun dato e quindi la stringa in risposta contiene soltanto il carattere 'N'.

'N'

7.2.3 Comando 'i' (informazioni)

Questo comando fornisce alcune indicazioni sullo stato del programma.

'i'

La risposta è una stringa di lunghezza massima 32 byte della forma "mode:xx samples:yy\n" con xx modalità attuale della macchina a stati e yy numero di campioni inviati nella modalità di acquisizione con time-break.

Per quanto riguarda la macchina a stati interna relativa all'acquisizione dei campioni sono possibili i seguenti valori:

0 lettura e memorizzazione continua dei campioni
1 scrittura valori offset calibrazione
2 scrittura valori gain calibrazione
3 attesa impulso di time-break
4 lettura e memorizzazione del numero di campioni programmato dopo il time-break

7.2.4 Comando 'c' (scrittura offset)

Questo comando è stato previsto per scrivere i valori di offset della calibrazione ma non è stato ancora implementato in quanto non influente per la definizione dei collegamenti tra le due schede.

'c'

7.2.5 Comando 'g' (scrittura gain)

Questo comando è stato previsto per scrivere i valori di gain della calibrazione ma non è stato ancora implementato in quanto non influente per la definizione dei collegamenti tra le due schede.

'g'

7.2.6 Comando 'p' (acquisizione con time-break)

Questo comando porta la macchina a stati di acquisizione nella modalità 3 che interrompe la memorizzazione dei campioni, azzera i puntatori dei buffer e rimane in attesa dell'arrivo del time-break. All'arrivo del time-break, che in questa versione è generato dalla pressione del tasto blu presente sulla

scheda, il sistema passa alla modalità 4 in cui, scartati i primi 31 campioni dovuti ai ritardo del filtro a fase lineare, memorizza il numero di campioni programmato e ritorna quindi in modalità 3.

7.2.7 Comando 'n' (acquisizione immediata)

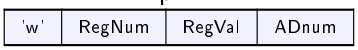
Questo comando simula l'arrivo del time-break, più esattamente esegue le stesse operazioni di inizializzazione del comando 'p' ma anziché passare alla modalità 3 salta direttamente alla 4 eseguendo quindi immediatamente la memorizzazione del numero di campioni programmato.

7.2.8 Comando 'f' (acquisizione continua)

Questo comando porta la macchina a stati di acquisizione nella modalità 0, ossia quella di default, in cui vengono memorizzati tutti i campioni.

7.2.9 Comando 'w' (scrittura registro)

Questo comando permette di scrivere un particolare registro di un AD converter.



Il comando prevede tre parametri, il numero del registro da scrivere (RegNum) che può avere un valore tra 0 e 11, il valore da scrivere nel registro (RegVal) ed il numero dell'AD converter (ADnum) che può avere un valore tra 1 e 6.

7.2.10 Comando 's' (numero campioni)

Questo comando definisce il numero di campioni da acquisire nella modalità 4.

Il comando prevede due parametri che rappresentano il byte più significativo (HIval) ed il byte meno significativo (LOval) del numero da memorizzare che sarà quindi pari a 256 * HIval + LOval.

7.2.11 Comando 'x' (reset AD)

Questo comando attiva la linea di reset collegata a tutti gli AD converter. L'impulso ha una durata di 5 microsecondi.

7.3 Gestione SPI

Il collegamento SPI è utilizzato per la lettura dei dati campionati e per l'accesso ai registri degli AD converter (ADS1282) presenti sulla scheda GEOPAD. Nel capitolo 4.5.2 sono già state evidenziate le caratteristiche fisiche del collegamento SPI, ossia frequenza del clock, polarità e fase. Anche nello sviluppo del software di comunicazione sono da tenere in considerazione le specifiche dell'ADS1282 che di fatto definiscono le caratteristiche del collegamento. Tutte le routine sono contenute nel file SPI.c.

Il protocollo di comunicazione prevede due modalità di funzionamento abbastanza diverse, la prima riguarda l'acquisizione del dato e viene richiesta dall'AD converter quando è disponibile un nuovo dato, la seconda invece è richiesta da un sistema esterno mediante il protocollo ethernet discusso nel capitolo 7.2 e consiste nell'accesso ai registri interni dell'AD converter.

L'AD converter è configurato per acquisire il dato nella modalità di default (*Read Data Continuous*). Quando risulta disponibile un nuovo dato l'ADS1282 abbassa la linea Data Ready ($\overline{\text{DRDY}}$) così come indicato nella figura 7.1, questo segnale viene utilizzato per generare l'interrupt che chiama la procedura SPI_GetSample in cui avviene la lettura e la memorizzazione del dato.

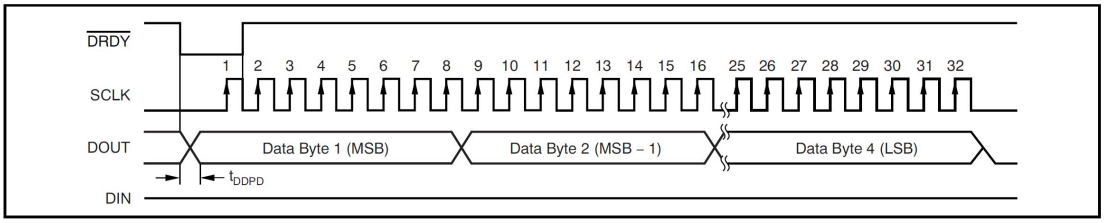


Figura 7.1: ADS1282 Read Data Continuous

La lettura e scrittura dei registri avviene invece tramite le due routine SPI_RegsRead e SPI_RegWrite. In questa modalità, che richiede ovviamente il blocco dell'acquisizione, ci sono alcune restrizioni sulla comunicazione da parte dell'ADS1282, in particolare è richiesto un ritardo minimo tra la lettura e scrittura di due byte che nella configurazione attuale è pari a circa sei microsecondi.

8. Connessioni

La figura 8.1 riporta tutti i segnali della scheda NUCLEO-H743ZI2 connessi di default al *Zio Connector*. Questo connettore è in realtà composto da quattro connettori separati (CN7, CN8, CN9 e CN10) ed i segnali connessi sono a 3,3V.

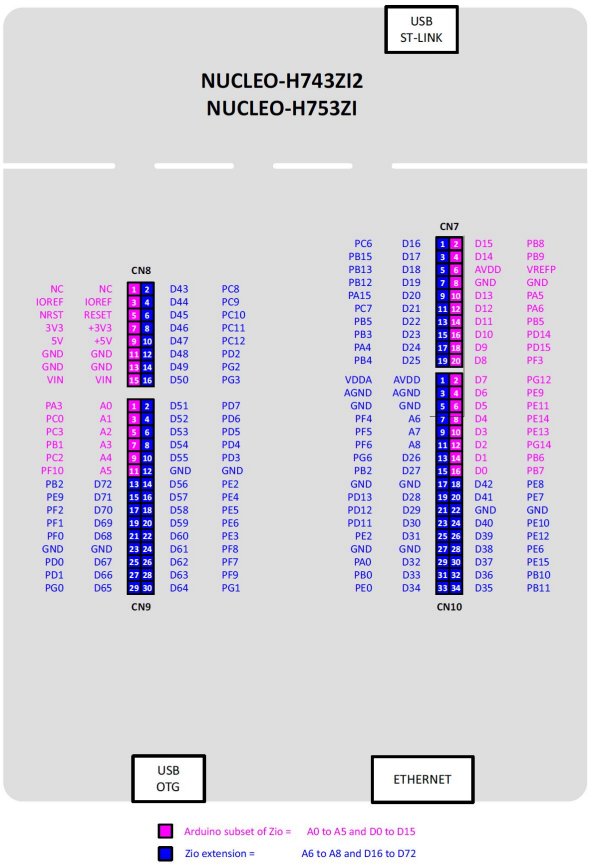


Figura 8.1: NUCLEO-H743ZI2 *Zio Connector*

Dopo aver sviluppato e testato il software si è ottenuta la seguente tabella per le connessioni tra la NUCLEO-H743ZI2 e la nuova scheda GEOPAD.

Tabella 8.1: Associazione segnali a pin *Zio connector*

Segnale		**pin micro**	**connettore**	**pin**
POWER	VIN	VIN	CN8	15
	GND	GND	CN8	13
	GND	GND	CN8	11
DRDY		PG6	CN10	13
SYNC		PG0	CN9	29
RESET		PC8	CN8	2
SPI1	SCK	PA5	CN7	10
	MISO	PA6	CN7	12
	MOSI	PD7	CN9	2
SPI2	SCK	PB10	CN10	32
	MISO	PC2	CN9	9
	MOSI	PC3	CN9	5
SPI3	SCK	PC10	CN8	6
	MISO	PC11	CN8	8
	MOSI	PC12	CN8	10
SPI4	SCK	PE2	CN9	14
	MISO	PE5	CN9	18
	MOSI	PE6	CN9	20
SPI5	SCK	PF7	CN9	26
	MISO	PF8	CN9	24
	MOSI	PF9	CN9	28
SPI6	SCK	PB3	CN7	15
	MISO	PG12	CN10	2
	MOSI	PG14	CN10	12
MFLAG1		PB9	CN7	4
MFLAG2		PB8	CN7	2
MFLAG3		PB5	CN7	14
MFLAG4		PB15	CN7	3
MFLAG5		PC9	CN8	4
MFLAG6		PD2	CN8	12
TB1		PE0	CN10	33
TB2		PB11	CN10	34

La scelta dei pin per i segnali non vincolati dalla struttura interna del microcontrollore (RESET, DRDY, SYNC, MFLAG1..MFLAG6, TB1 e TB2) è stata effettuata tenendo conto dei problemi di sbrogliaggio della nuova scheda GEOPAD. In pratica, visto l'elevato numero di pin disponibili, si sono scelti quelli in cui la pista di collegamento risultava più semplice da realizzare.

Utilizzando i dati della tabella 8.1 lo schematico dei collegamenti della nuova scheda GEOPAD è risultato quello di figura 8.2, dove i connettori J1, J2, J3 e J4 corrispondono ai connettori CN8, CN9, CN7 e CN10 della scheda NUCLEO-H743ZI2.

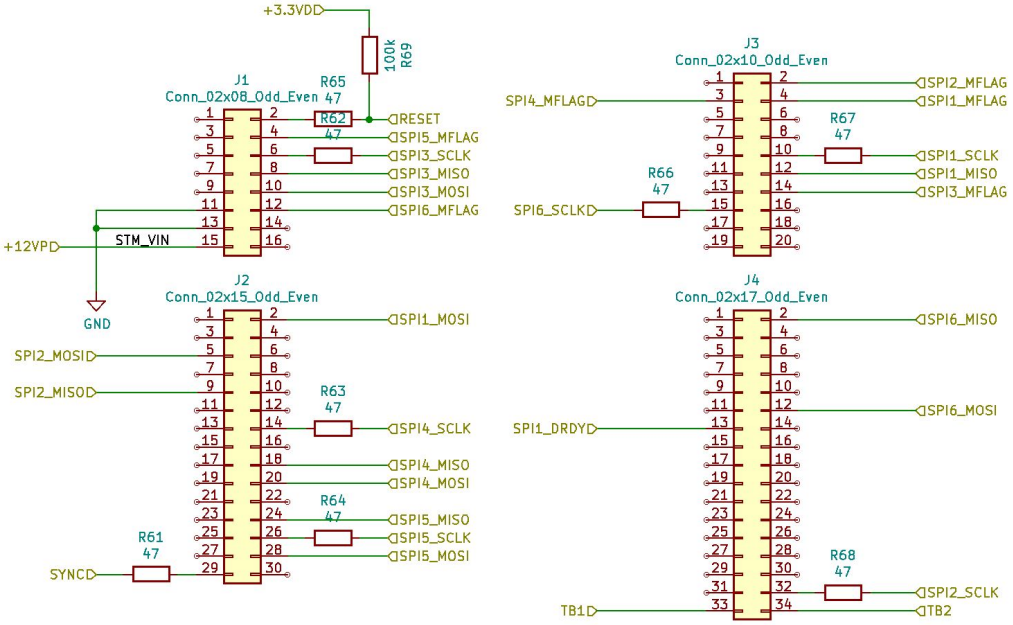


Figura 8.2: Connessioni tra GEOPAD e *Zio Connector*

Bibliografia

[1] *ADS1282 High-Resolution Analog-To-Digital Converter*. SBAS418I. Revised march 2015. Texas Instruments. Set. 2007.

[2] *STM32H742xI/G STM32H743xI/G - 32-bit Arm® Cortex®-M7 480MHz MCUs, up to 2MB Flash, up to 1MB RAM, 46 com. and analog interfaces - Datasheet*. DS12110. Rev. 7. STMicroelectronics. Apr. 2019.

[3] *UM1713 - User Manual - Developing applications on STM32Cube with LwIP TCP/IP stack*. DocID025731. Rev. 4. STMicroelectronics. Mag. 2015.

[4] *UM1727 - User Manual - Getting started with STM32 Nucleo board software development tools*. DocID025840. Rev. 5. STMicroelectronics. Gen. 2016.

[5] *UM1974 - User Manual - STM32 Nucleo-144 boards*. DocID028599. Rev. 7. STMicroelectronics. Dic. 2017.

[6] *UM2204 - User Manual - Getting started with STM32CubeH7 for STM32H7 Series*. UM2204. Rev. 7. STMicroelectronics. Apr. 2020.

[7] *UM2407 - User Manual - STM32H7 Nucleo-144 boards (MB1364)*. UM2407. Rev. 1. STMicroelectronics. Mar. 2019.

Elenco delle figure

Elenco delle tabelle

A. Sorgenti

A.1 main.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "lwip.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "SPI.h"
#include <stdio.h>
#include "stm32h7xx_nucleo_144.h"
#include "tcp_server.h"
#include "udp_echoserver.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define RANGE_12BITS    ((uint32_t) 4095)  // Max digital value with a full range of 12 bits
#define RANGE_16BITS    ((uint32_t)65535)  // Max digital value with a full range of 16 bits

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

```

```
41 /* Private variables ---------------------------------------------------------*/
42
43 DAC_HandleTypeDef hdac1;
44
45 SPI_HandleTypeDef hspi1;
46 SPI_HandleTypeDef hspi2;
47 SPI_HandleTypeDef hspi3;
48 SPI_HandleTypeDef hspi4;
49 SPI_HandleTypeDef hspi5;
50 SPI_HandleTypeDef hspi6;
51
52 TIM_HandleTypeDef htim2;
53 TIM_HandleTypeDef htim3;
54
55 UART_HandleTypeDef huart3;
56
57 /* USER CODE BEGIN PV */
58 uint32_t TIM3count = 0;
59
60 uint8_t aTxBuffer[BUFFERSIZE];       /* Buffer used for transmission */
61 uint8_t aRxBuffer[BUFFERSIZE];       /* Buffer used for reception */
62
63 /* transfer state */
64 __IO uint32_t wTransferState;
65
66 int mode;
67 int samplesent;
68 int samplestosend;
69 int bufin;
70 int bufout;
71 int bufind;
72 uint32_t bufcounter;
73 uint32_t bufc[4];            // for each group of buffers one counter value
74 uint8_t buf[6][4][4*256];    // for each ad (6 in total) 4 buffers of 256 samples (4 bytes)
75
76 /* USER CODE END PV */
77
78 /* Private function prototypes -----------------------------------------------*/
79 void SystemClock_Config(void);
80 static void MPU_Config(void);
81 static void MX_GPIO_Init(void);
82 static void MX_USART3_UART_Init(void);
83 static void MX_TIM3_Init(void);
84 static void MX_DAC1_Init(void);
85 static void MX_SPI1_Init(void);
86 static void MX_SPI2_Init(void);
87 static void MX_SPI3_Init(void);
88 static void MX_SPI4_Init(void);
89 static void MX_SPI5_Init(void);
90 static void MX_SPI6_Init(void);
91 static void MX_TIM2_Init(void);
92 /* USER CODE BEGIN PFP */
93 static void MPU_Config(void);
94 /* USER CODE END PFP */
95
96 /* Private user code ---------------------------------------------------------*/
97 /* USER CODE BEGIN 0 */
98
99 #ifdef __GNUC__
```

```
100   #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
101 #else
102   #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
103 #endif /* __GNUC__ */
104
105 PUTCHAR_PROTOTYPE
106 {
107   HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 0xFFFF);
108   return ch;
109 }
110
111 void delay_us (uint32_t us)
112 {
113     uint32_t h2time;
114
115     __HAL_TIM_SET_COUNTER(&htim2,0);     // set the counter value a 0
116     do                                   // wait for the counter to reach the us input in the par
117     {
118         h2time = __HAL_TIM_GET_COUNTER(&htim2);
119     } while( h2time < us );
120 }
121
122 /**
123   * @brief EXTI line detection callbacks
124   * @param GPIO_Pin: Specifies the pins connected EXTI line
125   * @retval None
126   */
127 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
128 {
129     if (GPIO_Pin == GPIO_PIN_6)
130     {
131         SPI_GetSample();
132     } else if(GPIO_Pin == GPIO_PIN_13)       // blu button
133     {
134         if( mode == 3 )
135         {
136             mode = 4;
137             samplesent = 0;
138             bufin = 0;
139             bufout = 0;
140             bufind = 0;
141             bufcounter = 0;
142         }
143     }
144 }
145
146 /**
147   * @brief  This function is executed in case of error occurrence.
148   * @param  None
149   * @retval None
150   */
151 void Timeout_Error_Handler(void)
152 {
153   while(1)
154   {
155     BSP_LED_On(LED3);
156     HAL_Delay(500);
157     BSP_LED_Off(LED3);
158     HAL_Delay(500);
```

```
159     }
160   }
161
162   /* USER CODE END 0 */
163
164   /**
165     * @brief  The application entry point.
166     * @retval int
167     */
168   int main(void)
169   {
170     /* USER CODE BEGIN 1 */
171     /* USER CODE END 1 */
172
173     /* MPU Configuration--------------------------------------------------------*/
174     MPU_Config();
175
176     /* Enable I-Cache---------------------------------------------------------*/
177     SCB_EnableICache();
178
179     /* Enable D-Cache---------------------------------------------------------*/
180     SCB_EnableDCache();
181
182     /* MCU Configuration--------------------------------------------------------*/
183
184     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
185     HAL_Init();
186
187     /* USER CODE BEGIN Init */
188
189     samplesent = 0;
190     samplestosend = 4096;
191     mode = 0;      // 0 == Continuous Data Read
192     bufin = 0;
193     bufout = 0;
194     bufind = 0;
195     bufcounter = 0;
196
197     /* USER CODE END Init */
198
199     /* Configure the system clock */
200     SystemClock_Config();
201
202     /* USER CODE BEGIN SysInit */
203
204     /* USER CODE END SysInit */
205
206     /* Initialize all configured peripherals */
207     MX_GPIO_Init();
208     MX_USART3_UART_Init();
209     MX_LWIP_Init();
210     MX_TIM3_Init();
211     MX_DAC1_Init();
212     MX_SPI1_Init();
213     MX_SPI2_Init();
214     MX_SPI3_Init();
215     MX_SPI4_Init();
216     MX_SPI5_Init();
217     MX_SPI6_Init();
```

```
218   MX_TIM2_Init();
219   /* USER CODE BEGIN 2 */
220   BSP_LED_Init(LED2);
221   udp_echoserver_init();
222   tcp_server_init();
223
224   if (HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, RANGE_12BITS/2) != HAL_OK)
225   {
226     Error_Handler();
227   }
228   if (HAL_DAC_Start(&hdac1, DAC_CHANNEL_1) != HAL_OK)
229   {
230     Error_Handler();
231   }
232
233   HAL_TIM_Base_Start(&htim2);        // start timer2 for 1 us counter
234
235   /* USER CODE END 2 */
236
237   /* Infinite loop */
238   /* USER CODE BEGIN WHILE */
239   while (1)
240   {
241     MX_LWIP_Process();
242     HAL_Delay(1);
243     /* USER CODE END WHILE */
244
245     /* USER CODE BEGIN 3 */
246   }
247   /* USER CODE END 3 */
248 }
249
250 /**
251   * @brief System Clock Configuration
252   * @retval None
253   */
254 void SystemClock_Config(void)
255 {
256   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
257   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
258   RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
259
260   /** Supply configuration update enable
261   */
262   HAL_PWREx_ConfigSupply(PWR_LDO_SUPPLY);
263   /** Configure the main internal regulator output voltage
264   */
265   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
266
267   while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}
268   /** Macro to configure the PLL clock source
269   */
270   __HAL_RCC_PLL_PLLSOURCE_CONFIG(RCC_PLLSOURCE_HSE);
271   /** Initializes the CPU, AHB and APB busses clocks
272   */
273   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
274   RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
275   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
276   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
```

```
277   RCC_OscInitStruct.PLL.PLLM = 1;
278   RCC_OscInitStruct.PLL.PLLN = 100;
279   RCC_OscInitStruct.PLL.PLLP = 2;
280   RCC_OscInitStruct.PLL.PLLQ = 2;
281   RCC_OscInitStruct.PLL.PLLR = 2;
282   RCC_OscInitStruct.PLL.PLLRGE = RCC_PLL1VCIRANGE_3;
283   RCC_OscInitStruct.PLL.PLLVCOSEL = RCC_PLL1VCOWIDE;
284   RCC_OscInitStruct.PLL.PLLFRACN = 0;
285   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
286   {
287     Error_Handler();
288   }
289   /** Initializes the CPU, AHB and APB busses clocks
290   */
291   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
292                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2
293                                 |RCC_CLOCKTYPE_D3PCLK1|RCC_CLOCKTYPE_D1PCLK1;
294   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
295   RCC_ClkInitStruct.SYSCLKDivider = RCC_SYSCLK_DIV1;
296   RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV2;
297   RCC_ClkInitStruct.APB3CLKDivider = RCC_APB3_DIV2;
298   RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV2;
299   RCC_ClkInitStruct.APB2CLKDivider = RCC_APB2_DIV2;
300   RCC_ClkInitStruct.APB4CLKDivider = RCC_APB4_DIV2;
301
302   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
303   {
304     Error_Handler();
305   }
306   PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART3|RCC_PERIPHCLK_SPI5
307                                 |RCC_PERIPHCLK_SPI4|RCC_PERIPHCLK_SPI3
308                                 |RCC_PERIPHCLK_SPI1|RCC_PERIPHCLK_SPI2
309                                 |RCC_PERIPHCLK_SPI6;
310   PeriphClkInitStruct.PLL2.PLL2M = 1;
311   PeriphClkInitStruct.PLL2.PLL2N = 50;
312   PeriphClkInitStruct.PLL2.PLL2P = 4;
313   PeriphClkInitStruct.PLL2.PLL2Q = 4;
314   PeriphClkInitStruct.PLL2.PLL2R = 2;
315   PeriphClkInitStruct.PLL2.PLL2RGE = RCC_PLL2VCIRANGE_3;
316   PeriphClkInitStruct.PLL2.PLL2VCOSEL = RCC_PLL2VCOWIDE;
317   PeriphClkInitStruct.PLL2.PLL2FRACN = 0;
318   PeriphClkInitStruct.Spi123ClockSelection = RCC_SPI123CLKSOURCE_PLL2;
319   PeriphClkInitStruct.Spi45ClockSelection = RCC_SPI45CLKSOURCE_D2PCLK1;
320   PeriphClkInitStruct.Usart234578ClockSelection = RCC_USART234578CLKSOURCE_D2PCLK1;
321   PeriphClkInitStruct.Spi6ClockSelection = RCC_SPI6CLKSOURCE_D3PCLK1;
322   if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
323   {
324     Error_Handler();
325   }
326 }
327
328 /**
329   * @brief DAC1 Initialization Function
330   * @param None
331   * @retval None
332   */
333 static void MX_DAC1_Init(void)
334 {
335
```

```
336   /* USER CODE BEGIN DAC1_Init 0 */
337
338   /* USER CODE END DAC1_Init 0 */
339
340   DAC_ChannelConfTypeDef sConfig = {0};
341
342   /* USER CODE BEGIN DAC1_Init 1 */
343
344   /* USER CODE END DAC1_Init 1 */
345   /** DAC Initialization
346   */
347   hdac1.Instance = DAC1;
348   if (HAL_DAC_Init(&hdac1) != HAL_OK)
349   {
350     Error_Handler();
351   }
352   /** DAC channel OUT1 config
353   */
354   sConfig.DAC_SampleAndHold = DAC_SAMPLEANDHOLD_DISABLE;
355   sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
356   sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
357   sConfig.DAC_ConnectOnChipPeripheral = DAC_CHIPCONNECT_DISABLE;
358   sConfig.DAC_UserTrimming = DAC_TRIMMING_FACTORY;
359   if (HAL_DAC_ConfigChannel(&hdac1, &sConfig, DAC_CHANNEL_1) != HAL_OK)
360   {
361     Error_Handler();
362   }
363   /* USER CODE BEGIN DAC1_Init 2 */
364
365   /* USER CODE END DAC1_Init 2 */
366
367 }
368
369 /**
370   * @brief SPI1 Initialization Function
371   * @param None
372   * @retval None
373   */
374 static void MX_SPI1_Init(void)
375 {
376
377   /* USER CODE BEGIN SPI1_Init 0 */
378
379   /* USER CODE END SPI1_Init 0 */
380
381   /* USER CODE BEGIN SPI1_Init 1 */
382
383   /* USER CODE END SPI1_Init 1 */
384   /* SPI1 parameter configuration*/
385   hspi1.Instance = SPI1;
386   hspi1.Init.Mode = SPI_MODE_MASTER;
387   hspi1.Init.Direction = SPI_DIRECTION_2LINES;
388   hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
389   hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
390   hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
391   hspi1.Init.NSS = SPI_NSS_SOFT;
392   hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
393   hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
394   hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
```

```
395   hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
396   hspi1.Init.CRCPolynomial = 0x0;
397   hspi1.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
398   hspi1.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
399   hspi1.Init.FifoThreshold = SPI_FIFO_THRESHOLD_01DATA;
400   hspi1.Init.TxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
401   hspi1.Init.RxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
402   hspi1.Init.MasterSSIdleness = SPI_MASTER_SS_IDLENESS_00CYCLE;
403   hspi1.Init.MasterInterDataIdleness = SPI_MASTER_INTERDATA_IDLENESS_00CYCLE;
404   hspi1.Init.MasterReceiverAutoSusp = SPI_MASTER_RX_AUTOSUSP_DISABLE;
405   hspi1.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;
406   hspi1.Init.IOSwap = SPI_IO_SWAP_DISABLE;
407   if (HAL_SPI_Init(&hspi1) != HAL_OK)
408   {
409     Error_Handler();
410   }
411   /* USER CODE BEGIN SPI1_Init 2 */
412
413   /* USER CODE END SPI1_Init 2 */
414
415 }
416
417 /**
418   * @brief SPI2 Initialization Function
419   * @param None
420   * @retval None
421   */
422 static void MX_SPI2_Init(void)
423 {
424
425   /* USER CODE BEGIN SPI2_Init 0 */
426
427   /* USER CODE END SPI2_Init 0 */
428
429   /* USER CODE BEGIN SPI2_Init 1 */
430
431   /* USER CODE END SPI2_Init 1 */
432   /* SPI2 parameter configuration*/
433   hspi2.Instance = SPI2;
434   hspi2.Init.Mode = SPI_MODE_MASTER;
435   hspi2.Init.Direction = SPI_DIRECTION_2LINES;
436   hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
437   hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
438   hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
439   hspi2.Init.NSS = SPI_NSS_SOFT;
440   hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
441   hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
442   hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
443   hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
444   hspi2.Init.CRCPolynomial = 0x0;
445   hspi2.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
446   hspi2.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
447   hspi2.Init.FifoThreshold = SPI_FIFO_THRESHOLD_01DATA;
448   hspi2.Init.TxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
449   hspi2.Init.RxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
450   hspi2.Init.MasterSSIdleness = SPI_MASTER_SS_IDLENESS_00CYCLE;
451   hspi2.Init.MasterInterDataIdleness = SPI_MASTER_INTERDATA_IDLENESS_00CYCLE;
452   hspi2.Init.MasterReceiverAutoSusp = SPI_MASTER_RX_AUTOSUSP_DISABLE;
453   hspi2.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;
```

```
454   hspi2.Init.IOSwap = SPI_IO_SWAP_DISABLE;
455   if (HAL_SPI_Init(&hspi2) != HAL_OK)
456   {
457     Error_Handler();
458   }
459   /* USER CODE BEGIN SPI2_Init 2 */
460
461   /* USER CODE END SPI2_Init 2 */
462
463 }
464
465 /**
466   * @brief SPI3 Initialization Function
467   * @param None
468   * @retval None
469   */
470 static void MX_SPI3_Init(void)
471 {
472
473   /* USER CODE BEGIN SPI3_Init 0 */
474
475   /* USER CODE END SPI3_Init 0 */
476
477   /* USER CODE BEGIN SPI3_Init 1 */
478
479   /* USER CODE END SPI3_Init 1 */
480   /* SPI3 parameter configuration*/
481   hspi3.Instance = SPI3;
482   hspi3.Init.Mode = SPI_MODE_MASTER;
483   hspi3.Init.Direction = SPI_DIRECTION_2LINES;
484   hspi3.Init.DataSize = SPI_DATASIZE_8BIT;
485   hspi3.Init.CLKPolarity = SPI_POLARITY_LOW;
486   hspi3.Init.CLKPhase = SPI_PHASE_1EDGE;
487   hspi3.Init.NSS = SPI_NSS_SOFT;
488   hspi3.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
489   hspi3.Init.FirstBit = SPI_FIRSTBIT_MSB;
490   hspi3.Init.TIMode = SPI_TIMODE_DISABLE;
491   hspi3.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
492   hspi3.Init.CRCPolynomial = 0x0;
493   hspi3.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
494   hspi3.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
495   hspi3.Init.FifoThreshold = SPI_FIFO_THRESHOLD_01DATA;
496   hspi3.Init.TxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
497   hspi3.Init.RxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
498   hspi3.Init.MasterSSIdleness = SPI_MASTER_SS_IDLENESS_00CYCLE;
499   hspi3.Init.MasterInterDataIdleness = SPI_MASTER_INTERDATA_IDLENESS_00CYCLE;
500   hspi3.Init.MasterReceiverAutoSusp = SPI_MASTER_RX_AUTOSUSP_DISABLE;
501   hspi3.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;
502   hspi3.Init.IOSwap = SPI_IO_SWAP_DISABLE;
503   if (HAL_SPI_Init(&hspi3) != HAL_OK)
504   {
505     Error_Handler();
506   }
507   /* USER CODE BEGIN SPI3_Init 2 */
508
509   /* USER CODE END SPI3_Init 2 */
510
511 }
512
```

```
513 /**
514   * @brief SPI4 Initialization Function
515   * @param None
516   * @retval None
517   */
518 static void MX_SPI4_Init(void)
519 {
520
521   /* USER CODE BEGIN SPI4_Init 0 */
522
523   /* USER CODE END SPI4_Init 0 */
524
525   /* USER CODE BEGIN SPI4_Init 1 */
526
527   /* USER CODE END SPI4_Init 1 */
528   /* SPI4 parameter configuration*/
529   hspi4.Instance = SPI4;
530   hspi4.Init.Mode = SPI_MODE_MASTER;
531   hspi4.Init.Direction = SPI_DIRECTION_2LINES;
532   hspi4.Init.DataSize = SPI_DATASIZE_8BIT;
533   hspi4.Init.CLKPolarity = SPI_POLARITY_LOW;
534   hspi4.Init.CLKPhase = SPI_PHASE_1EDGE;
535   hspi4.Init.NSS = SPI_NSS_SOFT;
536   hspi4.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
537   hspi4.Init.FirstBit = SPI_FIRSTBIT_MSB;
538   hspi4.Init.TIMode = SPI_TIMODE_DISABLE;
539   hspi4.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
540   hspi4.Init.CRCPolynomial = 0x0;
541   hspi4.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
542   hspi4.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
543   hspi4.Init.FifoThreshold = SPI_FIFO_THRESHOLD_01DATA;
544   hspi4.Init.TxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
545   hspi4.Init.RxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
546   hspi4.Init.MasterSSIdleness = SPI_MASTER_SS_IDLENESS_00CYCLE;
547   hspi4.Init.MasterInterDataIdleness = SPI_MASTER_INTERDATA_IDLENESS_00CYCLE;
548   hspi4.Init.MasterReceiverAutoSusp = SPI_MASTER_RX_AUTOSUSP_DISABLE;
549   hspi4.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;
550   hspi4.Init.IOSwap = SPI_IO_SWAP_DISABLE;
551   if (HAL_SPI_Init(&hspi4) != HAL_OK)
552   {
553     Error_Handler();
554   }
555   /* USER CODE BEGIN SPI4_Init 2 */
556
557   /* USER CODE END SPI4_Init 2 */
558
559 }
560
561 /**
562   * @brief SPI5 Initialization Function
563   * @param None
564   * @retval None
565   */
566 static void MX_SPI5_Init(void)
567 {
568
569   /* USER CODE BEGIN SPI5_Init 0 */
570
571   /* USER CODE END SPI5_Init 0 */
```

```
572
573   /* USER CODE BEGIN SPI5_Init 1 */
574
575   /* USER CODE END SPI5_Init 1 */
576   /* SPI5 parameter configuration*/
577   hspi5.Instance = SPI5;
578   hspi5.Init.Mode = SPI_MODE_MASTER;
579   hspi5.Init.Direction = SPI_DIRECTION_2LINES;
580   hspi5.Init.DataSize = SPI_DATASIZE_8BIT;
581   hspi5.Init.CLKPolarity = SPI_POLARITY_LOW;
582   hspi5.Init.CLKPhase = SPI_PHASE_1EDGE;
583   hspi5.Init.NSS = SPI_NSS_SOFT;
584   hspi5.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
585   hspi5.Init.FirstBit = SPI_FIRSTBIT_MSB;
586   hspi5.Init.TIMode = SPI_TIMODE_DISABLE;
587   hspi5.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
588   hspi5.Init.CRCPolynomial = 0x0;
589   hspi5.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
590   hspi5.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
591   hspi5.Init.FifoThreshold = SPI_FIFO_THRESHOLD_01DATA;
592   hspi5.Init.TxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
593   hspi5.Init.RxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
594   hspi5.Init.MasterSSIdleness = SPI_MASTER_SS_IDLENESS_00CYCLE;
595   hspi5.Init.MasterInterDataIdleness = SPI_MASTER_INTERDATA_IDLENESS_00CYCLE;
596   hspi5.Init.MasterReceiverAutoSusp = SPI_MASTER_RX_AUTOSUSP_DISABLE;
597   hspi5.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;
598   hspi5.Init.IOSwap = SPI_IO_SWAP_DISABLE;
599   if (HAL_SPI_Init(&hspi5) != HAL_OK)
600   {
601     Error_Handler();
602   }
603   /* USER CODE BEGIN SPI5_Init 2 */
604
605   /* USER CODE END SPI5_Init 2 */
606
607 }
608
609 /**
610   * @brief SPI6 Initialization Function
611   * @param None
612   * @retval None
613   */
614 static void MX_SPI6_Init(void)
615 {
616
617   /* USER CODE BEGIN SPI6_Init 0 */
618
619   /* USER CODE END SPI6_Init 0 */
620
621   /* USER CODE BEGIN SPI6_Init 1 */
622
623   /* USER CODE END SPI6_Init 1 */
624   /* SPI6 parameter configuration*/
625   hspi6.Instance = SPI6;
626   hspi6.Init.Mode = SPI_MODE_MASTER;
627   hspi6.Init.Direction = SPI_DIRECTION_2LINES;
628   hspi6.Init.DataSize = SPI_DATASIZE_8BIT;
629   hspi6.Init.CLKPolarity = SPI_POLARITY_LOW;
630   hspi6.Init.CLKPhase = SPI_PHASE_1EDGE;
```

```
631   hspi6.Init.NSS = SPI_NSS_SOFT;
632   hspi6.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
633   hspi6.Init.FirstBit = SPI_FIRSTBIT_MSB;
634   hspi6.Init.TIMode = SPI_TIMODE_DISABLE;
635   hspi6.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
636   hspi6.Init.CRCPolynomial = 0x0;
637   hspi6.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
638   hspi6.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
639   hspi6.Init.FifoThreshold = SPI_FIFO_THRESHOLD_01DATA;
640   hspi6.Init.TxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
641   hspi6.Init.RxCRCInitializationPattern = SPI_CRC_INITIALIZATION_ALL_ZERO_PATTERN;
642   hspi6.Init.MasterSSIdleness = SPI_MASTER_SS_IDLENESS_00CYCLE;
643   hspi6.Init.MasterInterDataIdleness = SPI_MASTER_INTERDATA_IDLENESS_00CYCLE;
644   hspi6.Init.MasterReceiverAutoSusp = SPI_MASTER_RX_AUTOSUSP_DISABLE;
645   hspi6.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;
646   hspi6.Init.IOSwap = SPI_IO_SWAP_DISABLE;
647   if (HAL_SPI_Init(&hspi6) != HAL_OK)
648   {
649     Error_Handler();
650   }
651   /* USER CODE BEGIN SPI6_Init 2 */
652
653   /* USER CODE END SPI6_Init 2 */
654
655 }
656
657 /**
658   * @brief TIM2 Initialization Function
659   * @param None
660   * @retval None
661   */
662 static void MX_TIM2_Init(void)
663 {
664
665   /* USER CODE BEGIN TIM2_Init 0 */
666
667   /* USER CODE END TIM2_Init 0 */
668
669   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
670   TIM_MasterConfigTypeDef sMasterConfig = {0};
671
672   /* USER CODE BEGIN TIM2_Init 1 */
673
674   /* USER CODE END TIM2_Init 1 */
675   htim2.Instance = TIM2;
676   htim2.Init.Prescaler = 200-1;
677   htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
678   htim2.Init.Period = 0xffffffff-1;
679   htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
680   htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
681   if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
682   {
683     Error_Handler();
684   }
685   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
686   if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
687   {
688     Error_Handler();
689   }
```

```
690   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
691   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
692   if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
693   {
694     Error_Handler();
695   }
696   /* USER CODE BEGIN TIM2_Init 2 */
697
698   /* USER CODE END TIM2_Init 2 */
699
700 }
701
702 /**
703   * @brief TIM3 Initialization Function
704   * @param None
705   * @retval None
706   */
707 static void MX_TIM3_Init(void)
708 {
709
710   /* USER CODE BEGIN TIM3_Init 0 */
711
712   /* USER CODE END TIM3_Init 0 */
713
714   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
715   TIM_MasterConfigTypeDef sMasterConfig = {0};
716
717   /* USER CODE BEGIN TIM3_Init 1 */
718
719   /* USER CODE END TIM3_Init 1 */
720   htim3.Instance = TIM3;
721   htim3.Init.Prescaler = 19999;
722   htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
723   htim3.Init.Period = 9;
724   htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
725   htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
726   if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
727   {
728     Error_Handler();
729   }
730   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
731   if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
732   {
733     Error_Handler();
734   }
735   sMasterConfig.MasterOutputTrigger = TIM_TRGO_ENABLE;
736   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
737   if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
738   {
739     Error_Handler();
740   }
741   /* USER CODE BEGIN TIM3_Init 2 */
742   if (HAL_TIM_Base_Start_IT(&htim3) != HAL_OK)
743    {
744     /* Starting Error */
745     Error_Handler();
746    }
747
748   /* USER CODE END TIM3_Init 2 */
```

```
749
750 }
751
752 /**
753   * @brief USART3 Initialization Function
754   * @param None
755   * @retval None
756   */
757 static void MX_USART3_UART_Init(void)
758 {
759
760   /* USER CODE BEGIN USART3_Init 0 */
761
762   /* USER CODE END USART3_Init 0 */
763
764   /* USER CODE BEGIN USART3_Init 1 */
765
766   /* USER CODE END USART3_Init 1 */
767   huart3.Instance = USART3;
768   huart3.Init.BaudRate = 115200;
769   huart3.Init.WordLength = UART_WORDLENGTH_8B;
770   huart3.Init.StopBits = UART_STOPBITS_1;
771   huart3.Init.Parity = UART_PARITY_NONE;
772   huart3.Init.Mode = UART_MODE_TX_RX;
773   huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
774   huart3.Init.OverSampling = UART_OVERSAMPLING_16;
775   huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
776   huart3.Init.ClockPrescaler = UART_PRESCALER_DIV1;
777   huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
778   if (HAL_UART_Init(&huart3) != HAL_OK)
779   {
780     Error_Handler();
781   }
782   if (HAL_UARTEx_SetTxFifoThreshold(&huart3, UART_TXFIFO_THRESHOLD_1_8) != HAL_OK)
783   {
784     Error_Handler();
785   }
786   if (HAL_UARTEx_SetRxFifoThreshold(&huart3, UART_RXFIFO_THRESHOLD_1_8) != HAL_OK)
787   {
788     Error_Handler();
789   }
790   if (HAL_UARTEx_DisableFifoMode(&huart3) != HAL_OK)
791   {
792     Error_Handler();
793   }
794   /* USER CODE BEGIN USART3_Init 2 */
795
796   /* USER CODE END USART3_Init 2 */
797
798 }
799
800 /**
801   * @brief GPIO Initialization Function
802   * @param None
803   * @retval None
804   */
805 static void MX_GPIO_Init(void)
806 {
807   GPIO_InitTypeDef GPIO_InitStruct = {0};
```

```
808
809     /* GPIO Ports Clock Enable */
810     __HAL_RCC_GPIOE_CLK_ENABLE();
811     __HAL_RCC_GPIOC_CLK_ENABLE();
812     __HAL_RCC_GPIOF_CLK_ENABLE();
813     __HAL_RCC_GPIOH_CLK_ENABLE();
814     __HAL_RCC_GPIOA_CLK_ENABLE();
815     __HAL_RCC_GPIOB_CLK_ENABLE();
816     __HAL_RCC_GPIOG_CLK_ENABLE();
817     __HAL_RCC_GPIOD_CLK_ENABLE();
818
819     /*Configure GPIO pin Output Level */
820     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_14, GPIO_PIN_RESET);
821
822     /*Configure GPIO pin Output Level */
823     HAL_GPIO_WritePin(SYNC_GPIO_Port, SYNC_Pin, GPIO_PIN_RESET);
824
825     /*Configure GPIO pin Output Level */
826     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_RESET);
827
828     /*Configure GPIO pin Output Level */
829     HAL_GPIO_WritePin(ADRESET_GPIO_Port, ADRESET_Pin, GPIO_PIN_RESET);
830
831     /*Configure GPIO pin Output Level */
832     HAL_GPIO_WritePin(GPIOE, GPIO_PIN_1, GPIO_PIN_RESET);
833
834     /*Configure GPIO pin : PC13 */
835     GPIO_InitStruct.Pin = GPIO_PIN_13;
836     GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
837     GPIO_InitStruct.Pull = GPIO_NOPULL;
838     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
839
840     /*Configure GPIO pins : PB0 PB14 */
841     GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_14;
842     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
843     GPIO_InitStruct.Pull = GPIO_NOPULL;
844     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
845     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
846
847     /*Configure GPIO pin : SYNC_Pin */
848     GPIO_InitStruct.Pin = SYNC_Pin;
849     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
850     GPIO_InitStruct.Pull = GPIO_NOPULL;
851     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
852     HAL_GPIO_Init(SYNC_GPIO_Port, &GPIO_InitStruct);
853
854     /*Configure GPIO pin : TB2_Pin */
855     GPIO_InitStruct.Pin = TB2_Pin;
856     GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
857     GPIO_InitStruct.Pull = GPIO_NOPULL;
858     HAL_GPIO_Init(TB2_GPIO_Port, &GPIO_InitStruct);
859
860     /*Configure GPIO pins : MFLAG4_Pin MFLAG3_Pin MFLAG2_Pin MFLAG1_Pin */
861     GPIO_InitStruct.Pin = MFLAG4_Pin|MFLAG3_Pin|MFLAG2_Pin|MFLAG1_Pin;
862     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
863     GPIO_InitStruct.Pull = GPIO_NOPULL;
864     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
865
866     /*Configure GPIO pin : PD10 */
```

```
867     GPIO_InitStruct.Pin = GPIO_PIN_10;
868     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
869     GPIO_InitStruct.Pull = GPIO_NOPULL;
870     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
871     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
872 
873     /*Configure GPIO pin : DRDY1_Pin */
874     GPIO_InitStruct.Pin = DRDY1_Pin;
875     GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
876     GPIO_InitStruct.Pull = GPIO_NOPULL;
877     HAL_GPIO_Init(DRDY1_GPIO_Port, &GPIO_InitStruct);
878 
879     /*Configure GPIO pin : ADRESET_Pin */
880     GPIO_InitStruct.Pin = ADRESET_Pin;
881     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
882     GPIO_InitStruct.Pull = GPIO_NOPULL;
883     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
884     HAL_GPIO_Init(ADRESET_GPIO_Port, &GPIO_InitStruct);
885 
886     /*Configure GPIO pin : MFLAG5_Pin */
887     GPIO_InitStruct.Pin = MFLAG5_Pin;
888     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
889     GPIO_InitStruct.Pull = GPIO_NOPULL;
890     HAL_GPIO_Init(MFLAG5_GPIO_Port, &GPIO_InitStruct);
891 
892     /*Configure GPIO pins : PA8 PA11 PA12 */
893     GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_11|GPIO_PIN_12;
894     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
895     GPIO_InitStruct.Pull = GPIO_NOPULL;
896     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
897     GPIO_InitStruct.Alternate = GPIO_AF10_OTG1_FS;
898     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
899 
900     /*Configure GPIO pin : MFLAG6_Pin */
901     GPIO_InitStruct.Pin = MFLAG6_Pin;
902     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
903     GPIO_InitStruct.Pull = GPIO_NOPULL;
904     HAL_GPIO_Init(MFLAG6_GPIO_Port, &GPIO_InitStruct);
905 
906     /*Configure GPIO pin : TB1_Pin */
907     GPIO_InitStruct.Pin = TB1_Pin;
908     GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
909     GPIO_InitStruct.Pull = GPIO_NOPULL;
910     HAL_GPIO_Init(TB1_GPIO_Port, &GPIO_InitStruct);
911 
912     /*Configure GPIO pin : PE1 */
913     GPIO_InitStruct.Pin = GPIO_PIN_1;
914     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
915     GPIO_InitStruct.Pull = GPIO_NOPULL;
916     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
917     HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
918 
919     /* EXTI interrupt init*/
920     HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
921     HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
922 
923     HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
924     HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
925 
```

```
926 }
927
928 /* USER CODE BEGIN 4 */
929 /* USER CODE END 4 */
930
931 /* MPU Configuration */
932
933 void MPU_Config(void)
934 {
935   MPU_Region_InitTypeDef MPU_InitStruct = {0};
936
937   /* Disables the MPU */
938   HAL_MPU_Disable();
939   /** Initializes and configures the Region and the memory to be protected
940   */
941   MPU_InitStruct.Enable = MPU_REGION_ENABLE;
942   MPU_InitStruct.Number = MPU_REGION_NUMBER0;
943   MPU_InitStruct.BaseAddress = 0x30040000;
944   MPU_InitStruct.Size = MPU_REGION_SIZE_256B;
945   MPU_InitStruct.SubRegionDisable = 0x0;
946   MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
947   MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
948   MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;
949   MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
950   MPU_InitStruct.IsCacheable = MPU_ACCESS_NOT_CACHEABLE;
951   MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;
952
953   HAL_MPU_ConfigRegion(&MPU_InitStruct);
954   /** Initializes and configures the Region and the memory to be protected
955   */
956   MPU_InitStruct.Enable = MPU_REGION_ENABLE;
957   MPU_InitStruct.Number = MPU_REGION_NUMBER1;
958   MPU_InitStruct.BaseAddress = 0x30044000;
959   MPU_InitStruct.Size = MPU_REGION_SIZE_16KB;
960   MPU_InitStruct.SubRegionDisable = 0x0;
961   MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
962   MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
963   MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;
964   MPU_InitStruct.IsShareable = MPU_ACCESS_SHAREABLE;
965   MPU_InitStruct.IsCacheable = MPU_ACCESS_NOT_CACHEABLE;
966   MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
967
968   HAL_MPU_ConfigRegion(&MPU_InitStruct);
969   /* Enables the MPU */
970   HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
971
972 }
973 /**
974   * @brief  Period elapsed callback in non blocking mode
975   * @note   This function is called  when TIM1 interrupt took place, inside
976   * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
977   * a global variable "uwTick" used as application time base.
978   * @param  htim : TIM handle
979   * @retval None
980   */
981 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
982 {
983   /* USER CODE BEGIN Callback 0 */
984
```

```
985   if (htim->Instance == TIM3) {
986       if(TIM3count++ >= 1000)
987       {
988           TIM3count = 0;
989           BSP_LED_Toggle(LED2);
990       }
991   }
992 
993   /* USER CODE END Callback 0 */
994   if (htim->Instance == TIM1) {
995     HAL_IncTick();
996   }
997   /* USER CODE BEGIN Callback 1 */
998 
999   /* USER CODE END Callback 1 */
1000 }
1001 
1002 /**
1003   * @brief  This function is executed in case of error occurrence.
1004   * @retval None
1005   */
1006 void Error_Handler(void)
1007 {
1008   /* USER CODE BEGIN Error_Handler_Debug */
1009   /* User can add his own implementation to report the HAL error return state */
1010 
1011   /* USER CODE END Error_Handler_Debug */
1012 }
1013 
1014 #ifdef  USE_FULL_ASSERT
1015 /**
1016   * @brief  Reports the name of the source file and the source line number
1017   *         where the assert_param error has occurred.
1018   * @param  file: pointer to the source file name
1019   * @param  line: assert_param error line source number
1020   * @retval None
1021   */
1022 void assert_failed(uint8_t *file, uint32_t line)
1023 {
1024   /* USER CODE BEGIN 6 */
1025   /* User can add his own implementation to report the file name and line number,
1026      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
1027   /* USER CODE END 6 */
1028 }
1029 #endif /* USE_FULL_ASSERT */
1030 
1031 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.2 main.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32h7xx_hal.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

enum {
  TRANSFER_WAIT,
  TRANSFER_COMPLETE,
  TRANSFER_ERROR
};

/* Size of SPI Tx/Rx buffer */
#define BUFFERSIZE  1024
/* UDP local connection port */
#define UDP_PORT 1111

/* USER CODE END EC */
```

```
57
58 /* Exported macro ------------------------------------------------------------*/
59 /* USER CODE BEGIN EM */
60
61 /* USER CODE END EM */
62
63 /* Exported functions prototypes ---------------------------------------------*/
64 void Error_Handler(void);
65
66 /* USER CODE BEGIN EFP */
67 extern uint32_t buffer[10000][6];
68 extern int buff_top;
69 extern int buff_bottom;
70 extern SPI_HandleTypeDef hspi1;
71 extern SPI_HandleTypeDef hspi2;
72 extern SPI_HandleTypeDef hspi3;
73 extern SPI_HandleTypeDef hspi4;
74 extern SPI_HandleTypeDef hspi5;
75 extern SPI_HandleTypeDef hspi6;
76 extern uint8_t aTxBuffer[BUFFERSIZE];
77 extern uint8_t aRxBuffer[BUFFERSIZE];
78 extern __IO uint32_t wTransferState;
79
80 extern int mode;
81 extern int samplesent;
82 extern int samplestosend;
83 extern int bufin,bufout,bufind;
84 extern uint32_t bufcounter;
85 extern uint32_t bufc[4];              // for each group of buffers one counter value
86 extern uint8_t buf[6][4][4*256];      // for each ad (6 in total) 4 buffers of 256 readings (32 bit sampl
87
88 void Timeout_Error_Handler(void);
89 void delay_us (uint32_t us);
90
91 /* USER CODE END EFP */
92
93 /* Private defines -----------------------------------------------------------*/
94 #define SYNC_Pin GPIO_PIN_0
95 #define SYNC_GPIO_Port GPIOG
96 #define TB2_Pin GPIO_PIN_11
97 #define TB2_GPIO_Port GPIOB
98 #define TB2_EXTI_IRQn EXTI15_10_IRQn
99 #define MFLAG4_Pin GPIO_PIN_15
100 #define MFLAG4_GPIO_Port GPIOB
101 #define DRDY1_Pin GPIO_PIN_6
102 #define DRDY1_GPIO_Port GPIOG
103 #define DRDY1_EXTI_IRQn EXTI9_5_IRQn
104 #define ADRESET_Pin GPIO_PIN_8
105 #define ADRESET_GPIO_Port GPIOC
106 #define MFLAG5_Pin GPIO_PIN_9
107 #define MFLAG5_GPIO_Port GPIOC
108 #define MFLAG6_Pin GPIO_PIN_2
109 #define MFLAG6_GPIO_Port GPIOD
110 #define MFLAG3_Pin GPIO_PIN_5
111 #define MFLAG3_GPIO_Port GPIOB
112 #define MFLAG2_Pin GPIO_PIN_8
113 #define MFLAG2_GPIO_Port GPIOB
114 #define MFLAG1_Pin GPIO_PIN_9
115 #define MFLAG1_GPIO_Port GPIOB
```

```
116 #define TB1_Pin GPIO_PIN_0
117 #define TB1_GPIO_Port GPIOE
118 /* USER CODE BEGIN Private defines */
119
120 /* USER CODE END Private defines */
121
122 #ifdef __cplusplus
123 }
124 #endif
125
126 #endif /* __MAIN_H */
127
128 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.3 lwip.c

```
/**
 ******************************************************************************
  * File Name          : LWIP.c
  * Description        : This file provides initialization code for LWIP
  *                      middleWare.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "lwip.h"
#include "lwip/init.h"
#include "lwip/netif.h"
#if defined ( __CC_ARM )  /* MDK ARM Compiler */
#include "lwip/sio.h"
#endif /* MDK ARM Compiler */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
/* Private function prototypes -----------------------------------------------*/
/* ETH Variables initialization ----------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* Variables Initialization */
struct netif gnetif;
ip4_addr_t ipaddr;
ip4_addr_t netmask;
ip4_addr_t gw;
uint8_t IP_ADDRESS[4];
uint8_t NETMASK_ADDRESS[4];
uint8_t GATEWAY_ADDRESS[4];

/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/**
  * LwIP initialization function
  */
void MX_LWIP_Init(void)
{
```

```
57    /* IP addresses initialization */
58        IP_ADDRESS[0] = 192;
59        IP_ADDRESS[1] = 168;
60        IP_ADDRESS[2] = 1;
61        IP_ADDRESS[3] = 11;
62        NETMASK_ADDRESS[0] = 255;
63        NETMASK_ADDRESS[1] = 255;
64        NETMASK_ADDRESS[2] = 255;
65        NETMASK_ADDRESS[3] = 0;
66        GATEWAY_ADDRESS[0] = 192;
67        GATEWAY_ADDRESS[1] = 168;
68        GATEWAY_ADDRESS[2] = 1;
69        GATEWAY_ADDRESS[3] = 254;
70
71    /* Initilialize the LwIP stack without RTOS */
72    lwip_init();
73
74    /* IP addresses initialization without DHCP (IPv4) */
75    IP4_ADDR(&ipaddr, IP_ADDRESS[0], IP_ADDRESS[1], IP_ADDRESS[2], IP_ADDRESS[3]);
76    IP4_ADDR(&netmask, NETMASK_ADDRESS[0], NETMASK_ADDRESS[1] , NETMASK_ADDRESS[2], NETMASK_ADDRES
77    IP4_ADDR(&gw, GATEWAY_ADDRESS[0], GATEWAY_ADDRESS[1], GATEWAY_ADDRESS[2], GATEWAY_ADDRESS[3]);
78
79    /* add the network interface (IPv4/IPv6) without RTOS */
80    netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &ethernetif_init, &ethernet_input);
81
82    /* Registers the default network interface */
83    netif_set_default(&gnetif);
84
85    if (netif_is_link_up(&gnetif))
86    {
87      /* When the netif is fully configured this function must be called */
88      netif_set_up(&gnetif);
89    }
90    else
91    {
92      /* When the netif link is down this function must be called */
93      netif_set_down(&gnetif);
94    }
95
96  /* USER CODE BEGIN 3 */
97
98  /* USER CODE END 3 */
99  }
100
101 #ifdef USE_OBSOLETE_USER_CODE_SECTION_4
102 /* Kept to help code migration. (See new 4_1, 4_2... sections) */
103 /* Avoid to use this user section which will become obsolete. */
104 /* USER CODE BEGIN 4 */
105 /* USER CODE END 4 */
106 #endif
107
108 /**
109  * ----------------------------------------------------------------------
110  * Function given to help user to continue LwIP Initialization
111  * Up to user to complete or change this function ...
112  * Up to user to call this function in main.c in while (1) of main(void)
113 *-----------------------------------------------------------------------
114  * Read a received packet from the Ethernet buffers
115  * Send it to the lwIP stack for handling
```

```
116   * Handle timeouts if LWIP_TIMERS is set and without RTOS
117   * Handle the llink status if LWIP_NETIF_LINK_CALLBACK is set and without RTOS
118   */
119  void MX_LWIP_Process(void)
120  {
121  /* USER CODE BEGIN 4_1 */
122  /* USER CODE END 4_1 */
123    ethernetif_input(&gnetif);
124
125  /* USER CODE BEGIN 4_2 */
126  /* USER CODE END 4_2 */
127    /* Handle timeouts */
128    sys_check_timeouts();
129
130  /* USER CODE BEGIN 4_3 */
131  #if LWIP_NETIF_LINK_CALLBACK
132      Ethernet_Link_Periodic_Handle(&gnetif);
133  #endif
134  /* USER CODE END 4_3 */
135  }
136
137  #if defined ( __CC_ARM )  /* MDK ARM Compiler */
138  /**
139   * Opens a serial device for communication.
140   *
141   * @param devnum device number
142   * @return handle to serial device if successful, NULL otherwise
143   */
144  sio_fd_t sio_open(u8_t devnum)
145  {
146    sio_fd_t sd;
147
148  /* USER CODE BEGIN 7 */
149    sd = 0; // dummy code
150  /* USER CODE END 7 */
151
152    return sd;
153  }
154
155  /**
156   * Sends a single character to the serial device.
157   *
158   * @param c character to send
159   * @param fd serial device handle
160   *
161   * @note This function will block until the character can be sent.
162   */
163  void sio_send(u8_t c, sio_fd_t fd)
164  {
165  /* USER CODE BEGIN 8 */
166  /* USER CODE END 8 */
167  }
168
169  /**
170   * Reads from the serial device.
171   *
172   * @param fd serial device handle
173   * @param data pointer to data buffer for receiving
174   * @param len maximum length (in bytes) of data to receive
```

```
175   * @return number of bytes actually received - may be 0 if aborted by sio_read_abort
176   *
177   * @note This function will block until data can be received. The blocking
178   * can be cancelled by calling sio_read_abort().
179   */
180  u32_t sio_read(sio_fd_t fd, u8_t *data, u32_t len)
181  {
182    u32_t recved_bytes;
183
184  /* USER CODE BEGIN 9 */
185    recved_bytes = 0; // dummy code
186  /* USER CODE END 9 */
187    return recved_bytes;
188  }
189
190  /**
191   * Tries to read from the serial device. Same as sio_read but returns
192   * immediately if no data is available and never blocks.
193   *
194   * @param fd serial device handle
195   * @param data pointer to data buffer for receiving
196   * @param len maximum length (in bytes) of data to receive
197   * @return number of bytes actually received
198   */
199  u32_t sio_tryread(sio_fd_t fd, u8_t *data, u32_t len)
200  {
201    u32_t recved_bytes;
202
203  /* USER CODE BEGIN 10 */
204    recved_bytes = 0; // dummy code
205  /* USER CODE END 10 */
206    return recved_bytes;
207  }
208  #endif /* MDK ARM Compiler */
209
210  /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.4 lwip.h

```
/**
  ******************************************************************************
  * File Name          : LWIP.h
  * Description        : This file provides code for the configuration
  *                      of the LWIP.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************

  */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __mx_lwip_H
#define __mx_lwip_H
#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "lwip/opt.h"
#include "lwip/mem.h"
#include "lwip/memp.h"
#include "netif/etharp.h"
#include "lwip/dhcp.h"
#include "lwip/netif.h"
#include "lwip/timeouts.h"
#include "ethernetif.h"

/* Includes for RTOS ---------------------------------------------------------*/
#if WITH_RTOS
#include "lwip/tcpip.h"
#endif /* WITH_RTOS */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/* Global Variables ----------------------------------------------------------*/
extern ETH_HandleTypeDef heth;

/* LWIP init function */
void MX_LWIP_Init(void);

#if !WITH_RTOS
/* USER CODE BEGIN 1 */
/* Function defined in lwip.c to:
 *   - Read a received packet from the Ethernet buffers
 *   - Send it to the lwIP stack for handling
```

```
57   *   - Handle timeouts if NO_SYS_NO_TIMERS not set
58   */
59  void MX_LWIP_Process(void);
60
61  /* USER CODE END 1 */
62  #endif /* WITH_RTOS */
63
64  #ifdef __cplusplus
65  }
66  #endif
67  #endif /*__ mx_lwip_H */
68
69  /**
70    * @}
71    */
72
73  /**
74    * @}
75    */
76
77  /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.5 lwipopts.h

```

/**
  ******************************************************************************
  * File Name          : lwipopts.h
  * Description        : This file overrides LwIP stack default configuration
  *                      done in opt.h file.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __LWIPOPTS__H__
#define __LWIPOPTS__H__

#include "main.h"

/*----------------------------------------------------------------------------*/
/* Current version of LwIP supported by CubeMx: 2.0.3 -*/
/*----------------------------------------------------------------------------*/

/* Within 'USER CODE' section, code will be kept by default at each generation */
/* USER CODE BEGIN 0 */
#define LWIP_DEBUG 1
/* USER CODE END 0 */

#ifdef __cplusplus
 extern "C" {
#endif

/* STM32CubeMX Specific Parameters (not defined in opt.h) --------------------*/
/* Parameters set in STM32CubeMX LwIP Configuration GUI -*/
/*------ WITH_RTOS disabled (Since FREERTOS is not set) ------*/
#define WITH_RTOS 0
/*------ CHECKSUM_BY_HARDWARE disabled ------*/
#define CHECKSUM_BY_HARDWARE 0
/*----------------------------------------------------------------------------*/

/* LwIP Stack Parameters (modified compared to initialization value in opt.h) -*/
/* Parameters set in STM32CubeMX LwIP Configuration GUI -*/
/*------ Value in opt.h for NO_SYS: 0 ------*/
#define NO_SYS 1
/*------ Value in opt.h for SYS_LIGHTWEIGHT_PROT: 1 ------*/
#define SYS_LIGHTWEIGHT_PROT 0
/*------ Value in opt.h for MEM_ALIGNMENT: 1 ------*/
#define MEM_ALIGNMENT 4
/*------ Default Value for MEM_SIZE: 1600 ----*/
```

```
57 #define MEM_SIZE 8192
58 /*------ Default Value for H7 devices: 0x30044000 ------*/
59 #define LWIP_RAM_HEAP_POINTER 0x30044000
60 /*------ Value supported for H7 devices: 1 ------*/
61 #define LWIP_SUPPORT_CUSTOM_PBUF 1
62 /*------ Default Value for PBUF_POOL_BUFSIZE: 592 ----*/
63 #define PBUF_POOL_BUFSIZE 512
64 /*------ Value in opt.h for LWIP_ETHERNET: LWIP_ARP || PPPOE_SUPPORT -*/
65 #define LWIP_ETHERNET 1
66 /*------ Default Value for LWIP_BROADCAST_PING: 0 ----*/
67 #define LWIP_BROADCAST_PING 1
68 /*------ Default Value for LWIP_MULTICAST_PING: 0 ----*/
69 #define LWIP_MULTICAST_PING 1
70 /*------ Value in opt.h for LWIP_DNS_SECURE: (LWIP_DNS_SECURE_RAND_XID | LWIP_DNS_SECURE_NO_MULTI
71 #define LWIP_DNS_SECURE 7
72 /*------ Default Value for LWIP_UDPLITE: 0 ----*/
73 #define LWIP_UDPLITE 1
74 /*------ Value in opt.h for TCP_SND_QUEUELEN: (4*TCP_SND_BUF + (TCP_MSS - 1))/TCP_MSS ------*/
75 #define TCP_SND_QUEUELEN 9
76 /*------ Value in opt.h for TCP_SNDLOWAT: LWIP_MIN(LWIP_MAX(((TCP_SND_BUF)/2), (2 * TCP_MSS) + 1)
77 #define TCP_SNDLOWAT 1071
78 /*------ Value in opt.h for TCP_SNDQUEUELOWAT: LWIP_MAX(TCP_SND_QUEUELEN)/2, 5) -*/
79 #define TCP_SNDQUEUELOWAT 5
80 /*------ Value in opt.h for TCP_WND_UPDATE_THRESHOLD: LWIP_MIN(TCP_WND/4, TCP_MSS*4) ------*/
81 #define TCP_WND_UPDATE_THRESHOLD 536
82 /*------ Value in opt.h for LWIP_NETCONN: 1 ------*/
83 #define LWIP_NETCONN 0
84 /*------ Value in opt.h for LWIP_SOCKET: 1 ------*/
85 #define LWIP_SOCKET 0
86 /*------ Value in opt.h for RECV_BUFSIZE_DEFAULT: INT_MAX ------*/
87 #define RECV_BUFSIZE_DEFAULT 2000000000
88 /*------ Default Value for LWIP_STATS: 0 ----*/
89 #define LWIP_STATS 1
90 /*------ Default Value for LWIP_STATS_DISPLAY: 0 ----*/
91 #define LWIP_STATS_DISPLAY 1
92 /*------ Value in opt.h for MIB2_STATS: 0 or SNMP_LWIP_MIB2 ------*/
93 #define MIB2_STATS 0
94 /*------ Value in opt.h for CHECKSUM_GEN_IP: 1 ------*/
95 #define CHECKSUM_GEN_IP 0
96 /*------ Value in opt.h for CHECKSUM_GEN_UDP: 1 ------*/
97 #define CHECKSUM_GEN_UDP 0
98 /*------ Value in opt.h for CHECKSUM_GEN_TCP: 1 ------*/
99 #define CHECKSUM_GEN_TCP 0
100 /*------ Value in opt.h for CHECKSUM_GEN_ICMP: 1 ------*/
101 #define CHECKSUM_GEN_ICMP 0
102 /*------ Value in opt.h for CHECKSUM_GEN_ICMP6: 1 ------*/
103 #define CHECKSUM_GEN_ICMP6 0
104 /*------ Value in opt.h for CHECKSUM_CHECK_IP: 1 ------*/
105 #define CHECKSUM_CHECK_IP 0
106 /*------ Value in opt.h for CHECKSUM_CHECK_UDP: 1 ------*/
107 #define CHECKSUM_CHECK_UDP 0
108 /*------ Value in opt.h for CHECKSUM_CHECK_TCP: 1 ------*/
109 #define CHECKSUM_CHECK_TCP 0
110 /*------ Value in opt.h for CHECKSUM_CHECK_ICMP: 1 ------*/
111 #define CHECKSUM_CHECK_ICMP 0
112 /*------ Value in opt.h for CHECKSUM_CHECK_ICMP6: 1 ------*/
113 #define CHECKSUM_CHECK_ICMP6 0
114 /*------ Default Value for TCPIP_DEBUG: LWIP_DBG_OFF ----*/
115 #define TCPIP_DEBUG LWIP_DBG_ON
```

```
116 /*-----------------------------------------------------------------------------*/
117 /* USER CODE BEGIN 1 */
118
119 /* USER CODE END 1 */
120
121 #ifdef __cplusplus
122 }
123 #endif
124 #endif /*__LWIPOPTS__H__ */
125
126 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.6 lwippools.h

```
/**
 ******************************************************************************
  * File Name          : lwippools.h
  * Description        : This file provides initialization code for LWIP
  *                      middleWare.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */

 /*****************************************************************************
  * If MEMP_USE_CUSTOM_POOLS option is enabled:
  *  - include this lwippools.h file that defines additional pools beyond the
  *    "standard" ones required by lwIP.
  *  - make sure you have lwippools.h in your include path.
  *****************************************************************************/

/* MEMP_USE_CUSTOM_POOLS is enabled => This file is required by LwIP */

#ifdef __cplusplus
 extern "C" {
#endif

/* USER CODE BEGIN 0 */
/* Warning 1: The following code is only given as an example */
/* You can use this example code by uncommenting it */
/* ---------------- EXAMPLE of CODE ----------------------------*/
/* Define three pools with sizes 256, 512, and 1512 bytes */
/*
#if MEM_USE_POOLS
LWIP_MALLOC_MEMPOOL_START
LWIP_MALLOC_MEMPOOL(20, 256)
LWIP_MALLOC_MEMPOOL(10, 512)
LWIP_MALLOC_MEMPOOL(5, 1512)
LWIP_MALLOC_MEMPOOL_END
#endif
*/
/* ---------------- END of EXAMPLE of CODE ---------------------*/

/* USER CODE END 0 */

#ifdef __cplusplus
}
#endif

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.7 ethernetif.c

```
/**
  ******************************************************************************
  * File Name          : ethernetif.c
  * Description        : This file provides code for the configuration
  *                      of the ethernetif.c MiddleWare.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "lwip/opt.h"
#include "lwip/mem.h"
#include "lwip/memp.h"
#include "lwip/timeouts.h"
#include "netif/ethernet.h"
#include "netif/etharp.h"
#include "lwip/ethip6.h"
#include "ethernetif.h"
#include "lan8742.h"
#include <string.h>

/* Within 'USER CODE' section, code will be kept by default at each generation */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/* Private define ------------------------------------------------------------*/

/* Network interface name */
#define IFNAME0 's'
#define IFNAME1 't'

/* ETH Setting  */
#define ETH_RX_BUFFER_SIZE                     ( 1536UL )
#define ETH_DMA_TRANSMIT_TIMEOUT               ( 20U )

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* Private variables ---------------------------------------------------------*/
/*
@Note: This interface is implemented to operate in zero-copy mode only:
        - Rx buffers are allocated statically and passed directly to the LwIP stack
          they will return back to DMA after been processed by the stack.
```

```
57           - Tx Buffers will be allocated from LwIP stack memory heap,
58             then passed to ETH HAL driver.
59
60   @Notes:
61     1.a. ETH DMA Rx descriptors must be contiguous, the default count is 4,
62          to customize it please redefine ETH_RX_DESC_CNT in ETH GUI (Rx Descriptor Length)
63          so that updated value will be generated in stm32xxxx_hal_conf.h
64     1.b. ETH DMA Tx descriptors must be contiguous, the default count is 4,
65          to customize it please redefine ETH_TX_DESC_CNT in ETH GUI (Tx Descriptor Length)
66          so that updated value will be generated in stm32xxxx_hal_conf.h
67
68     2.a. Rx Buffers number must be between ETH_RX_DESC_CNT and 2*ETH_RX_DESC_CNT
69     2.b. Rx Buffers must have the same size: ETH_RX_BUFFER_SIZE, this value must
70          passed to ETH DMA in the init field (heth.Init.RxBuffLen)
71   */
72
73   #if defined ( __ICCARM__ ) /*!< IAR Compiler */
74
75   #pragma location=0x30040000
76   ETH_DMADescTypeDef  DMARxDscrTab[ETH_RX_DESC_CNT]; /* Ethernet Rx DMA Descriptors */
77   #pragma location=0x30040060
78   ETH_DMADescTypeDef  DMATxDscrTab[ETH_TX_DESC_CNT]; /* Ethernet Tx DMA Descriptors */
79   #pragma location=0x30040200
80   uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_RX_BUFFER_SIZE]; /* Ethernet Receive Buffers */
81
82   #elif defined ( __CC_ARM )  /* MDK ARM Compiler */
83
84   __attribute__((at(0x30040000))) ETH_DMADescTypeDef  DMARxDscrTab[ETH_RX_DESC_CNT]; /* Ethernet R
85   __attribute__((at(0x30040060))) ETH_DMADescTypeDef  DMATxDscrTab[ETH_TX_DESC_CNT]; /* Ethernet T
86   __attribute__((at(0x30040200))) uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_RX_BUFFER_SIZE]; /* Etherne
87
88   #elif defined ( __GNUC__ ) /* GNU Compiler */
89
90   ETH_DMADescTypeDef DMARxDscrTab[ETH_RX_DESC_CNT] __attribute__((section(".RxDecripSection"))); /
91   ETH_DMADescTypeDef DMATxDscrTab[ETH_TX_DESC_CNT] __attribute__((section(".TxDecripSection")));
     /* Ethernet Tx DMA Descriptors */
92   uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_RX_BUFFER_SIZE] __attribute__((section(".RxArraySection")))
93
94   #endif
95
96   /* USER CODE BEGIN 2 */
97
98   /* USER CODE END 2 */
99
100  /* Global Ethernet handle */
101  ETH_HandleTypeDef heth;
102  ETH_TxPacketConfig TxConfig;
103
104  /* Memory Pool Declaration */
105  LWIP_MEMPOOL_DECLARE(RX_POOL, 10, sizeof(struct pbuf_custom), "Zero-copy RX PBUF pool");
106
107  /* Private function prototypes -----------------------------------------------*/
108  int32_t ETH_PHY_IO_Init(void);
109  int32_t ETH_PHY_IO_DeInit (void);
110  int32_t ETH_PHY_IO_ReadReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t *pRegVal);
111  int32_t ETH_PHY_IO_WriteReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t RegVal);
112  int32_t ETH_PHY_IO_GetTick(void);
113
114  lan8742_Object_t LAN8742;
```

```
115 lan8742_IOCtx_t  LAN8742_IOCtx = {ETH_PHY_IO_Init,
116                                   ETH_PHY_IO_DeInit,
117                                   ETH_PHY_IO_WriteReg,
118                                   ETH_PHY_IO_ReadReg,
119                                   ETH_PHY_IO_GetTick};
120
121 /* USER CODE BEGIN 3 */
122
123 /* USER CODE END 3 */
124
125 /* Private functions ---------------------------------------------------------*/
126 void pbuf_free_custom(struct pbuf *p);
127 void Error_Handler(void);
128
129 void HAL_ETH_MspInit(ETH_HandleTypeDef* ethHandle)
130 {
131   GPIO_InitTypeDef GPIO_InitStruct = {0};
132   if(ethHandle->Instance==ETH)
133   {
134   /* USER CODE BEGIN ETH_MspInit 0 */
135
136   /* USER CODE END ETH_MspInit 0 */
137     /* Enable Peripheral clock */
138     __HAL_RCC_ETH1MAC_CLK_ENABLE();
139     __HAL_RCC_ETH1TX_CLK_ENABLE();
140     __HAL_RCC_ETH1RX_CLK_ENABLE();
141
142     __HAL_RCC_GPIOC_CLK_ENABLE();
143     __HAL_RCC_GPIOA_CLK_ENABLE();
144     __HAL_RCC_GPIOB_CLK_ENABLE();
145     __HAL_RCC_GPIOG_CLK_ENABLE();
146     /**ETH GPIO Configuration
147     PC1     ------> ETH_MDC
148     PA1     ------> ETH_REF_CLK
149     PA2     ------> ETH_MDIO
150     PA7     ------> ETH_CRS_DV
151     PC4     ------> ETH_RXD0
152     PC5     ------> ETH_RXD1
153     PB13    ------> ETH_TXD1
154     PG11    ------> ETH_TX_EN
155     PG13    ------> ETH_TXD0
156     */
157     GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_4|GPIO_PIN_5;
158     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
159     GPIO_InitStruct.Pull = GPIO_NOPULL;
160     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
161     GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
162     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
163
164     GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_7;
165     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
166     GPIO_InitStruct.Pull = GPIO_NOPULL;
167     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
168     GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
169     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
170
171     GPIO_InitStruct.Pin = GPIO_PIN_13;
172     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
173     GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
174       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
175       GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
176       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
177
178       GPIO_InitStruct.Pin = GPIO_PIN_11|GPIO_PIN_13;
179       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
180       GPIO_InitStruct.Pull = GPIO_NOPULL;
181       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
182       GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
183       HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
184
185       /* Peripheral interrupt init */
186       HAL_NVIC_SetPriority(ETH_IRQn, 0, 0);
187       HAL_NVIC_EnableIRQ(ETH_IRQn);
188     /* USER CODE BEGIN ETH_MspInit 1 */
189     /* USER CODE END ETH_MspInit 1 */
190     }
191   }
192
193   void HAL_ETH_MspDeInit(ETH_HandleTypeDef* ethHandle)
194   {
195     if(ethHandle->Instance==ETH)
196     {
197     /* USER CODE BEGIN ETH_MspDeInit 0 */
198
199     /* USER CODE END ETH_MspDeInit 0 */
200       /* Disable Peripheral clock */
201       __HAL_RCC_ETH1MAC_CLK_DISABLE();
202       __HAL_RCC_ETH1TX_CLK_DISABLE();
203       __HAL_RCC_ETH1RX_CLK_DISABLE();
204
205       /**ETH GPIO Configuration
206       PC1     ------> ETH_MDC
207       PA1     ------> ETH_REF_CLK
208       PA2     ------> ETH_MDIO
209       PA7     ------> ETH_CRS_DV
210       PC4     ------> ETH_RXD0
211       PC5     ------> ETH_RXD1
212       PB13    ------> ETH_TXD1
213       PG11    ------> ETH_TX_EN
214       PG13    ------> ETH_TXD0
215       */
216       HAL_GPIO_DeInit(GPIOC, GPIO_PIN_1|GPIO_PIN_4|GPIO_PIN_5);
217
218       HAL_GPIO_DeInit(GPIOA, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_7);
219
220       HAL_GPIO_DeInit(GPIOB, GPIO_PIN_13);
221
222       HAL_GPIO_DeInit(GPIOG, GPIO_PIN_11|GPIO_PIN_13);
223
224       /* Peripheral interrupt Deinit*/
225       HAL_NVIC_DisableIRQ(ETH_IRQn);
226
227     /* USER CODE BEGIN ETH_MspDeInit 1 */
228
229     /* USER CODE END ETH_MspDeInit 1 */
230     }
231   }
232
```

```
233 /* USER CODE BEGIN 4 */
234
235 /* USER CODE END 4 */
236
237 /*******************************************************************************
238                        LL Driver Interface ( LwIP stack --> ETH)
239 *******************************************************************************/
240 /**
241  * @brief In this function, the hardware should be initialized.
242  * Called from ethernetif_init().
243  *
244  * @param netif the already initialized lwip network interface structure
245  *        for this ethernetif
246  */
247 static void low_level_init(struct netif *netif)
248 {
249   HAL_StatusTypeDef hal_eth_init_status;
250   uint32_t idx = 0;
251   /* Start ETH HAL Init */
252
253    uint8_t MACAddr[6] ;
254   heth.Instance = ETH;
255   MACAddr[0] = 0x00;
256   MACAddr[1] = 0x80;
257   MACAddr[2] = 0xE1;
258   MACAddr[3] = 0x00;
259   MACAddr[4] = 0x00;
260   MACAddr[5] = 0x00;
261   heth.Init.MACAddr = &MACAddr[0];
262   heth.Init.MediaInterface = HAL_ETH_RMII_MODE;
263   heth.Init.TxDesc = DMATxDscrTab;
264   heth.Init.RxDesc = DMARxDscrTab;
265   heth.Init.RxBuffLen = 1536;
266
267   /* USER CODE BEGIN MACADDRESS */
268
269   /* USER CODE END MACADDRESS */
270
271   hal_eth_init_status = HAL_ETH_Init(&heth);
272
273   memset(&TxConfig, 0 , sizeof(ETH_TxPacketConfig));
274   TxConfig.Attributes = ETH_TX_PACKETS_FEATURES_CSUM | ETH_TX_PACKETS_FEATURES_CRCPAD;
275   TxConfig.ChecksumCtrl = ETH_CHECKSUM_IPHDR_PAYLOAD_INSERT_PHDR_CALC;
276   TxConfig.CRCPadCtrl = ETH_CRC_PAD_INSERT;
277
278   /* End ETH HAL Init */
279
280   /* Initialize the RX POOL */
281   LWIP_MEMPOOL_INIT(RX_POOL);
282
283 #if LWIP_ARP || LWIP_ETHERNET
284
285   /* set MAC hardware address length */
286   netif->hwaddr_len = ETH_HWADDR_LEN;
287
288   /* set MAC hardware address */
289   netif->hwaddr[0] =  heth.Init.MACAddr[0];
290   netif->hwaddr[1] =  heth.Init.MACAddr[1];
291   netif->hwaddr[2] =  heth.Init.MACAddr[2];
```

```
292     netif->hwaddr[3] =  heth.Init.MACAddr[3];
293     netif->hwaddr[4] =  heth.Init.MACAddr[4];
294     netif->hwaddr[5] =  heth.Init.MACAddr[5];
295
296     /* maximum transfer unit */
297     netif->mtu = ETH_MAX_PAYLOAD;
298
299     /* Accept broadcast address and ARP traffic */
300     /* don't set NETIF_FLAG_ETHARP if this device is not an ethernet one */
301     #if LWIP_ARP
302       netif->flags |= NETIF_FLAG_BROADCAST | NETIF_FLAG_ETHARP;
303     #else
304       netif->flags |= NETIF_FLAG_BROADCAST;
305     #endif /* LWIP_ARP */
306
307     for(idx = 0; idx < ETH_RX_DESC_CNT; idx ++)
308     {
309       HAL_ETH_DescAssignMemory(&heth, idx, Rx_Buff[idx], NULL);
310     }
311
312   /* USER CODE BEGIN PHY_PRE_CONFIG */
313
314   /* USER CODE END PHY_PRE_CONFIG */
315     /* Set PHY IO functions */
316     LAN8742_RegisterBusIO(&LAN8742, &LAN8742_IOCtx);
317
318     /* Initialize the LAN8742 ETH PHY */
319     LAN8742_Init(&LAN8742);
320
321     if (hal_eth_init_status == HAL_OK)
322     {
323     /* Get link state */
324     ethernet_link_check_state(netif);
325     }
326     else
327     {
328       Error_Handler();
329     }
330   #endif /* LWIP_ARP || LWIP_ETHERNET */
331
332   /* USER CODE BEGIN LOW_LEVEL_INIT */
333
334   /* USER CODE END LOW_LEVEL_INIT */
335   }
336
337   /**
338    * This function should do the actual transmission of the packet. The packet is
339    * contained in the pbuf that is passed to the function. This pbuf
340    * might be chained.
341    *
342    * @param netif the lwip network interface structure for this ethernetif
343    * @param p the MAC packet to send (e.g. IP packet including MAC addresses and type)
344    * @return ERR_OK if the packet could be sent
345    *         an err_t value if the packet couldn't be sent
346    *
347    * @note Returning ERR_MEM here if a DMA queue of your MAC is full can lead to
348    *       strange results. You might consider waiting for space in the DMA queue
349    *       to become availale since the stack doesn't retry to send a packet
350    *       dropped because of memory failure (except for the TCP timers).
```

```
351  */
352
353 static err_t low_level_output(struct netif *netif, struct pbuf *p)
354 {
355   uint32_t i=0, framelen = 0;
356   struct pbuf *q;
357   err_t errval = ERR_OK;
358   ETH_BufferTypeDef Txbuffer[ETH_TX_DESC_CNT];
359
360   memset(Txbuffer, 0 , ETH_TX_DESC_CNT*sizeof(ETH_BufferTypeDef));
361
362   for(q = p; q != NULL; q = q->next)
363   {
364     if(i >= ETH_TX_DESC_CNT)
365       return ERR_IF;
366
367     Txbuffer[i].buffer = q->payload;
368     Txbuffer[i].len = q->len;
369     framelen += q->len;
370
371     if(i>0)
372     {
373       Txbuffer[i-1].next = &Txbuffer[i];
374     }
375
376     if(q->next == NULL)
377     {
378       Txbuffer[i].next = NULL;
379     }
380
381     i++;
382   }
383
384   TxConfig.Length = framelen;
385   TxConfig.TxBuffer = Txbuffer;
386
387   /* Clean and Invalidate data cache */
388   SCB_CleanInvalidateDCache();
389
390   HAL_ETH_Transmit(&heth, &TxConfig, ETH_DMA_TRANSMIT_TIMEOUT);
391
392   return errval;
393 }
394
395 /**
396  * Should allocate a pbuf and transfer the bytes of the incoming
397  * packet from the interface into the pbuf.
398  *
399  * @param netif the lwip network interface structure for this ethernetif
400  * @return a pbuf filled with the received packet (including MAC header)
401  *         NULL on memory error
402    */
403 static struct pbuf * low_level_input(struct netif *netif)
404 {
405   struct pbuf *p = NULL;
406   ETH_BufferTypeDef RxBuff;
407   uint32_t framelength = 0;
408   struct pbuf_custom* custom_pbuf;
409
```

```
410
411   if (HAL_ETH_IsRxDataAvailable(&heth))
412   {
413     HAL_ETH_GetRxDataBuffer(&heth, &RxBuff);
414     HAL_ETH_GetRxDataLength(&heth, &framelength);
415
416     /* Build Rx descriptor to be ready for next data reception */
417     HAL_ETH_BuildRxDescriptors(&heth);
418
419 #if !defined(DUAL_CORE) || defined(CORE_CM7)
420     /* Invalidate data cache for ETH Rx Buffers */
421     SCB_InvalidateDCache_by_Addr((uint32_t *)RxBuff.buffer, framelength);
422 #endif
423
424     custom_pbuf  = (struct pbuf_custom*)LWIP_MEMPOOL_ALLOC(RX_POOL);
425     custom_pbuf->custom_free_function = pbuf_free_custom;
426
427     p = pbuf_alloced_custom(PBUF_RAW, framelength, PBUF_REF, custom_pbuf, RxBuff.buffer, ETH_RX_
428
429     return p;
430   }
431   else
432   {
433     return NULL;
434   }
435 }
436
437 /**
438  * This function should be called when a packet is ready to be read
439  * from the interface. It uses the function low_level_input() that
440  * should handle the actual reception of bytes from the network
441  * interface. Then the type of the received packet is determined and
442  * the appropriate input function is called.
443  *
444  * @param netif the lwip network interface structure for this ethernetif
445  */
446 void ethernetif_input(struct netif *netif)
447 {
448   err_t err;
449   struct pbuf *p;
450
451   /* move received packet into a new pbuf */
452   p = low_level_input(netif);
453
454   /* no packet could be read, silently ignore this */
455   if (p == NULL) return;
456
457   /* entry point to the LwIP stack */
458   err = netif->input(p, netif);
459
460   if (err != ERR_OK)
461   {
462     LWIP_DEBUGF(NETIF_DEBUG, ("ethernetif_input: IP input error\n"));
463     pbuf_free(p);
464     p = NULL;
465   }
466
467 }
468
```

```
469 #if !LWIP_ARP
470 /**
471  * This function has to be completed by user in case of ARP OFF.
472  *
473  * @param netif the lwip network interface structure for this ethernetif
474  * @return ERR_OK if ...
475  */
476 static err_t low_level_output_arp_off(struct netif *netif, struct pbuf *q, const ip4_addr_t *ipaddr)
477 {
478   err_t errval;
479   errval = ERR_OK;
480
481 /* USER CODE BEGIN 5 */
482
483 /* USER CODE END 5 */
484
485   return errval;
486
487 }
488 #endif /* LWIP_ARP */
489
490 /**
491  * Should be called at the beginning of the program to set up the
492  * network interface. It calls the function low_level_init() to do the
493  * actual setup of the hardware.
494  *
495  * This function should be passed as a parameter to netif_add().
496  *
497  * @param netif the lwip network interface structure for this ethernetif
498  * @return ERR_OK if the loopif is initialized
499  *         ERR_MEM if private data couldn't be allocated
500  *         any other err_t on error
501  */
502 err_t ethernetif_init(struct netif *netif)
503 {
504   LWIP_ASSERT("netif != NULL", (netif != NULL));
505
506 #if LWIP_NETIF_HOSTNAME
507   /* Initialize interface hostname */
508   netif->hostname = "lwip";
509 #endif /* LWIP_NETIF_HOSTNAME */
510
511   netif->name[0] = IFNAME0;
512   netif->name[1] = IFNAME1;
513   /* We directly use etharp_output() here to save a function call.
514    * You can instead declare your own function an call etharp_output()
515    * from it if you have to do some checks before sending (e.g. if link
516    * is available...) */
517
518 #if LWIP_IPV4
519 #if LWIP_ARP || LWIP_ETHERNET
520 #if LWIP_ARP
521   netif->output = etharp_output;
522 #else
523   /* The user should write ist own code in low_level_output_arp_off function */
524   netif->output = low_level_output_arp_off;
525 #endif /* LWIP_ARP */
526 #endif /* LWIP_ARP || LWIP_ETHERNET */
527 #endif /* LWIP_IPV4 */
```

```
528
529 #if LWIP_IPV6
530   netif->output_ip6 = ethip6_output;
531 #endif /* LWIP_IPV6 */
532
533   netif->linkoutput = low_level_output;
534
535   /* initialize the hardware */
536   low_level_init(netif);
537
538   return ERR_OK;
539 }
540
541 /**
542   * @brief  Custom Rx pbuf free callback
543   * @param  pbuf: pbuf to be freed
544   * @retval None
545   */
546 void pbuf_free_custom(struct pbuf *p)
547 {
548   struct pbuf_custom* custom_pbuf = (struct pbuf_custom*)p;
549
550 #if !defined(DUAL_CORE) || defined(CORE_CM7)
551   /* Invalidate data cache: lwIP and/or application may have written into buffer */
552   SCB_InvalidateDCache_by_Addr((uint32_t *)p->payload, p->tot_len);
553 #endif
554
555   LWIP_MEMPOOL_FREE(RX_POOL, custom_pbuf);
556 }
557
558 /* USER CODE BEGIN 6 */
559
560 /**
561 * @brief  Returns the current time in milliseconds
562 *         when LWIP_TIMERS == 1 and NO_SYS == 1
563 * @param  None
564 * @retval Current Time value
565 */
566 u32_t sys_jiffies(void)
567 {
568   return HAL_GetTick();
569 }
570
571 /**
572 * @brief  Returns the current time in milliseconds
573 *         when LWIP_TIMERS == 1 and NO_SYS == 1
574 * @param  None
575 * @retval Current Time value
576 */
577 u32_t sys_now(void)
578 {
579   return HAL_GetTick();
580 }
581
582 /* USER CODE END 6 */
583
584 /*******************************************************************************
585                        PHI IO Functions
586 *******************************************************************************/
```

```
587 /**
588   * @brief  Initializes the MDIO interface GPIO and clocks.
589   * @param  None
590   * @retval 0 if OK, -1 if ERROR
591   */
592 int32_t ETH_PHY_IO_Init(void)
593 {
594   /* We assume that MDIO GPIO configuration is already done
595      in the ETH_MspInit() else it should be done here
596   */
597
598   /* Configure the MDIO Clock */
599   HAL_ETH_SetMDIOClockRange(&heth);
600
601   return 0;
602 }
603
604 /**
605   * @brief  De-Initializes the MDIO interface .
606   * @param  None
607   * @retval 0 if OK, -1 if ERROR
608   */
609 int32_t ETH_PHY_IO_DeInit (void)
610 {
611   return 0;
612 }
613
614 /**
615   * @brief  Read a PHY register through the MDIO interface.
616   * @param  DevAddr: PHY port address
617   * @param  RegAddr: PHY register address
618   * @param  pRegVal: pointer to hold the register value
619   * @retval 0 if OK -1 if Error
620   */
621 int32_t ETH_PHY_IO_ReadReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t *pRegVal)
622 {
623   if(HAL_ETH_ReadPHYRegister(&heth, DevAddr, RegAddr, pRegVal) != HAL_OK)
624   {
625     return -1;
626   }
627
628   return 0;
629 }
630
631 /**
632   * @brief  Write a value to a PHY register through the MDIO interface.
633   * @param  DevAddr: PHY port address
634   * @param  RegAddr: PHY register address
635   * @param  RegVal: Value to be written
636   * @retval 0 if OK -1 if Error
637   */
638 int32_t ETH_PHY_IO_WriteReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t RegVal)
639 {
640   if(HAL_ETH_WritePHYRegister(&heth, DevAddr, RegAddr, RegVal) != HAL_OK)
641   {
642     return -1;
643   }
644
645   return 0;
```

```
646 }
647
648 /**
649   * @brief  Get the time in millisecons used for internal PHY driver process.
650   * @retval Time value
651   */
652 int32_t ETH_PHY_IO_GetTick(void)
653 {
654   return HAL_GetTick();
655 }
656
657 /**
658   * @brief  Check the ETH link state then update ETH driver and netif link accordingly.
659   * @param  argument: netif
660   * @retval None
661   */
662 void ethernet_link_check_state(struct netif *netif)
663 {
664   ETH_MACConfigTypeDef MACConf;
665   uint32_t PHYLinkState;
666   uint32_t linkchanged = 0, speed = 0, duplex =0;
667
668   PHYLinkState = LAN8742_GetLinkState(&LAN8742);
669
670   if(netif_is_link_up(netif) && (PHYLinkState <= LAN8742_STATUS_LINK_DOWN))
671   {
672     HAL_ETH_Stop(&heth);
673     netif_set_down(netif);
674     netif_set_link_down(netif);
675   }
676   else if(!netif_is_link_up(netif) && (PHYLinkState > LAN8742_STATUS_LINK_DOWN))
677   {
678     switch (PHYLinkState)
679     {
680     case LAN8742_STATUS_100MBITS_FULLDUPLEX:
681       duplex = ETH_FULLDUPLEX_MODE;
682       speed = ETH_SPEED_100M;
683       linkchanged = 1;
684       break;
685     case LAN8742_STATUS_100MBITS_HALFDUPLEX:
686       duplex = ETH_HALFDUPLEX_MODE;
687       speed = ETH_SPEED_100M;
688       linkchanged = 1;
689       break;
690     case LAN8742_STATUS_10MBITS_FULLDUPLEX:
691       duplex = ETH_FULLDUPLEX_MODE;
692       speed = ETH_SPEED_10M;
693       linkchanged = 1;
694       break;
695     case LAN8742_STATUS_10MBITS_HALFDUPLEX:
696       duplex = ETH_HALFDUPLEX_MODE;
697       speed = ETH_SPEED_10M;
698       linkchanged = 1;
699       break;
700     default:
701       break;
702     }
703
704     if(linkchanged)
```

```
705       {
706         /* Get MAC Config MAC */
707         HAL_ETH_GetMACConfig(&heth, &MACConf);
708         MACConf.DuplexMode = duplex;
709         MACConf.Speed = speed;
710         HAL_ETH_SetMACConfig(&heth, &MACConf);
711 
712         HAL_ETH_Start(&heth);
713         netif_set_up(netif);
714         netif_set_link_up(netif);
715       }
716     }
717 
718 }
719 /* USER CODE BEGIN 8 */
720 
721 /* USER CODE END 8 */
722 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.8 ethernetif.h

```
/**
  ******************************************************************************
  * File Name          : ethernetif.h
  * Description        : This file provides initialization code for LWIP
  *                      middleWare.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */


#ifndef __ETHERNETIF_H__
#define __ETHERNETIF_H__

#include "lwip/err.h"
#include "lwip/netif.h"

/* Within 'USER CODE' section, code will be kept by default at each generation */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/* Exported functions ------------------------------------------------------- */
err_t ethernetif_init(struct netif *netif);

void ethernetif_input(struct netif *netif);
void ethernet_link_check_state(struct netif *netif);

u32_t sys_jiffies(void);
u32_t sys_now(void);

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
#endif

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.9 SPI.c

```
/*
 * @file            : SPI.c
 *
 *  Created on: 17 feb 2020
 *      Author: Franco Zgauc
 *
 *
 */

#include <main.h>
#include <SPI.h>
#include "string.h"

void SPI_SendByte(SPI_HandleTypeDef *hspi, uint8_t ByteToSend)
{
    uint8_t TxBuffer[1] = {ByteToSend};

    switch(HAL_SPI_Transmit(hspi, TxBuffer, 1, 5000))
    {
        case HAL_OK:            // Communication is completed
            break;
        case HAL_TIMEOUT:       // A Timeout Occur
            Timeout_Error_Handler();
            break;
        case HAL_ERROR:         // An Error Occur
            Error_Handler();
            break;
        default:
            break;
    }
}

uint8_t SPI_ReceiveByte(SPI_HandleTypeDef *hspi)
{
    uint8_t RxBuffer[1] = {0};

    switch(HAL_SPI_Receive(hspi, RxBuffer, 1, 5000))
    {
        case HAL_OK:            // Communication is completed
            break;
        case HAL_TIMEOUT:       // A Timeout Occur
            Timeout_Error_Handler();
            break;
        case HAL_ERROR:         // An Error Occur
            Error_Handler();
            break;
        default:
            break;
    }
    return( RxBuffer[0] );
}

void SPI_RegsRead(SPI_HandleTypeDef *hspi, uint8_t *Registers)
{
    SPI_SendByte(hspi, AD_SDATAC);      // Stop read data continuous
    delay_us(SPI_DELAY);                // delay between bytes
```

```
57      SPI_SendByte(hspi, AD_RREG);          // read 11 REGs starting from ID (0h)
58      delay_us(SPI_DELAY);
59      SPI_SendByte(hspi, 10);               // number of registers - 1
60      delay_us(SPI_DELAY);
61
62      Registers[0] = SPI_ReceiveByte(hspi);   // id      - reset value x0h ID3,ID2,ID1,ID0,0,0,0,0
63      delay_us(SPI_DELAY);
64      Registers[1] = SPI_ReceiveByte(hspi);   // config0 - reset value 52h SYNC,1,DR2,DR1,DR0,PHS,
65      delay_us(SPI_DELAY);
66      Registers[2] = SPI_ReceiveByte(hspi);   // config1 - reset value 08h 0,MUX2,MUX1,MUX0,CHOP,P
67      delay_us(SPI_DELAY);
68      Registers[3] = SPI_ReceiveByte(hspi);   // hpf0    - reset value 32h HPF07,HPF06,HPF05,HPF04
69      delay_us(SPI_DELAY);
70      Registers[4] = SPI_ReceiveByte(hspi);   // hpf1    - reset value 03h HPF15,HPF14,HPF13,HPF12
71      delay_us(SPI_DELAY);
72      Registers[5] = SPI_ReceiveByte(hspi);   // ofc0    - reset value 00h OFC07,OFC06,OFC05,OFC04
73      delay_us(SPI_DELAY);
74      Registers[6] = SPI_ReceiveByte(hspi);   // ofc1    - reset value 00h OFC15,OFC14,OFC13,OFC12
75      delay_us(SPI_DELAY);
76      Registers[7] = SPI_ReceiveByte(hspi);   // ofc2    - reset value 00h OFC23,OFC22,OFC21,OFC20
77      delay_us(SPI_DELAY);
78      Registers[8] = SPI_ReceiveByte(hspi);   // fsc0    - reset value 00h FSC07,FSC06,FSC05,FSC04
79      delay_us(SPI_DELAY);
80      Registers[9] = SPI_ReceiveByte(hspi);   // fsc1    - reset value 00h FSC15,FSC14,FSC13,FSC12
81      delay_us(SPI_DELAY);
82      Registers[10] = SPI_ReceiveByte(hspi);  // fsc2    - reset value 40h FSC23,FSC22,FSC21,FSC20
83      delay_us(SPI_DELAY);
84
85      SPI_SendByte(hspi, AD_RDATAC);          //Read data continuous
86      delay_us(SPI_DELAY);
87  }
88
89  void SPI_RegWrite(SPI_HandleTypeDef *hspi, int regnum, int regval)
90  {
91      SPI_SendByte(hspi, AD_SDATAC);       // Stop read data continuous
92      delay_us(SPI_DELAY);                 // delay between bytes
93      SPI_SendByte(hspi, AD_WREG+regnum); // write regnum
94      delay_us(SPI_DELAY);
95      SPI_SendByte(hspi, 0);               // write 1 REG
96      delay_us(SPI_DELAY);
97      SPI_SendByte(hspi, regval);          // new register value
98      delay_us(SPI_DELAY);
99      SPI_SendByte(hspi, AD_RDATAC);       //Read data continuous
100     delay_us(SPI_DELAY);
101 }
102
103 void SPI_GetSample()
104 {
105     if( mode == 0 )
106     {
107         buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
108         buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
109         buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
110         buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
111         buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
112         buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
113         bufind++;
114         buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
115         buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
```

```
116             buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
117             buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
118             buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
119             buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
120             bufind++;
121             buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
122             buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
123             buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
124             buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
125             buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
126             buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
127             bufind++;
128             buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
129             buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
130             buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
131             buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
132             buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
133             buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
134             bufind++;
135             if(bufind >= 4*256)
136             {
137                 bufin = (bufin + 1) & 3;
138                 bufind = 0;
139                 bufcounter++;
140                 bufc[bufin] = bufcounter;
141             }
142         } else if ( mode == 1 )
143         {
144             SPI_SendByte(&hspi1, AD_SDATAC);     // Stop read data continuous
145             delay_us(SPI_DELAY);                 // delay between bytes
146             SPI_SendByte(&hspi1, AD_OFSCAL);     // Offset calibration
147             delay_us(SPI_DELAY);
148             SPI_SendByte(&hspi1, AD_RDATAC);     //Read data continuous
149             delay_us(SPI_DELAY);
150 
151             SPI_SendByte(&hspi2, AD_SDATAC);
152             delay_us(SPI_DELAY);
153             SPI_SendByte(&hspi2, AD_OFSCAL);
154             delay_us(SPI_DELAY);
155             SPI_SendByte(&hspi2, AD_RDATAC);
156             delay_us(SPI_DELAY);
157 
158             SPI_SendByte(&hspi3, AD_SDATAC);
159             delay_us(SPI_DELAY);
160             SPI_SendByte(&hspi3, AD_OFSCAL);
161             delay_us(SPI_DELAY);
162             SPI_SendByte(&hspi3, AD_RDATAC);
163             delay_us(SPI_DELAY);
164 
165             SPI_SendByte(&hspi4, AD_SDATAC);
166             delay_us(SPI_DELAY);
167             SPI_SendByte(&hspi4, AD_OFSCAL);
168             delay_us(SPI_DELAY);
169             SPI_SendByte(&hspi4, AD_RDATAC);
170             delay_us(SPI_DELAY);
171 
172             SPI_SendByte(&hspi5, AD_SDATAC);
173             delay_us(SPI_DELAY);
174             SPI_SendByte(&hspi5, AD_OFSCAL);
```

```
175             delay_us(SPI_DELAY);
176             SPI_SendByte(&hspi5, AD_RDATAC);
177             delay_us(SPI_DELAY);
178 
179             SPI_SendByte(&hspi6, AD_SDATAC);
180             delay_us(SPI_DELAY);
181             SPI_SendByte(&hspi6, AD_OFSCAL);
182             delay_us(SPI_DELAY);
183             SPI_SendByte(&hspi6, AD_RDATAC);
184             delay_us(SPI_DELAY);
185 
186             mode = 0;                               // 0 == Continuous Data Read
187         } else if ( mode == 2 )
188         {
189             SPI_SendByte(&hspi1, AD_SDATAC);        // Stop read data continuous
190             delay_us(SPI_DELAY);                    // delay between bytes
191             SPI_SendByte(&hspi1, AD_GANCAL);        // Gain calibration
192             delay_us(SPI_DELAY);
193             SPI_SendByte(&hspi1, AD_RDATAC);        //Read data continuous
194             delay_us(SPI_DELAY);
195 
196             SPI_SendByte(&hspi2, AD_SDATAC);
197             delay_us(SPI_DELAY);
198             SPI_SendByte(&hspi2, AD_GANCAL);
199             delay_us(SPI_DELAY);
200             SPI_SendByte(&hspi2, AD_RDATAC);
201             delay_us(SPI_DELAY);
202 
203             SPI_SendByte(&hspi3, AD_SDATAC);
204             delay_us(SPI_DELAY);
205             SPI_SendByte(&hspi3, AD_GANCAL);
206             delay_us(SPI_DELAY);
207             SPI_SendByte(&hspi3, AD_RDATAC);
208             delay_us(SPI_DELAY);
209 
210             SPI_SendByte(&hspi4, AD_SDATAC);
211             delay_us(SPI_DELAY);
212             SPI_SendByte(&hspi4, AD_GANCAL);
213             delay_us(SPI_DELAY);
214             SPI_SendByte(&hspi4, AD_RDATAC);
215             delay_us(SPI_DELAY);
216 
217             SPI_SendByte(&hspi5, AD_SDATAC);
218             delay_us(SPI_DELAY);
219             SPI_SendByte(&hspi5, AD_GANCAL);
220             delay_us(SPI_DELAY);
221             SPI_SendByte(&hspi5, AD_RDATAC);
222             delay_us(SPI_DELAY);
223 
224             SPI_SendByte(&hspi6, AD_SDATAC);
225             delay_us(SPI_DELAY);
226             SPI_SendByte(&hspi6, AD_GANCAL);
227             delay_us(SPI_DELAY);
228             SPI_SendByte(&hspi6, AD_RDATAC);
229             delay_us(SPI_DELAY);
230 
231             mode = 0;                               // 0 == Continuous Data Read
232         } else if ( mode == 3 )
233         {                                           // wait interrupt on nPULS
```

```
234     } else if ( mode == 4 )                 // get data after interrupt on nPULS
235     {
236         buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
237         buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
238         buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
239         buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
240         buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
241         buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
242         bufind++;
243         buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
244         buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
245         buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
246         buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
247         buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
248         buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
249         bufind++;
250         buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
251         buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
252         buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
253         buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
254         buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
255         buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
256         bufind++;
257         buf[0][bufin][bufind] = SPI_ReceiveByte(&hspi1);
258         buf[1][bufin][bufind] = SPI_ReceiveByte(&hspi2);
259         buf[2][bufin][bufind] = SPI_ReceiveByte(&hspi3);
260         buf[3][bufin][bufind] = SPI_ReceiveByte(&hspi4);
261         buf[4][bufin][bufind] = SPI_ReceiveByte(&hspi5);
262         buf[5][bufin][bufind] = SPI_ReceiveByte(&hspi6);
263         bufind++;
264         samplesent++;
265         if( samplesent == 31 )              // 31 samples filter delay if linear phase
266         {
267             bufind = 0;
268         }
269         if(bufind >= 4*256)
270         {
271             bufin = (bufin + 1) & 3;
272             bufind = 0;
273             bufcounter++;
274             bufc[bufin] = bufcounter;
275             if(samplesent >= samplestosend + 31)
276             {
277                 mode = 3;
278             }
279         }
280     }
281 }
```

A.10 SPI.h

```
/*
 * SPI.h
 *
 *  Created on: 17 feb 2020
 *      Author: Franco Zgauc
 */

#ifndef SPI_H_
#define SPI_H_

// ADS1282 Commands
#define AD_WAKEKUP 0x00  // Wake-up from Standby mode
#define AD_STANDBY 0x02  // Enter Standby mode
#define AD_SYNC    0x04  // Synchronize the A/D conversion
#define AD_RESET   0x06  // Reset registers to default values
#define AD_RDATAC  0x10  // Read data continuous
#define AD_SDATAC  0x11  // Stop read data continuous
#define AD_RDATA   0x12  // Read data by command
#define AD_RREG    0x20  // + 000rrrrr Read register(s) ad address rrrrr
#define AD_WREG    0x40  // + 000rrrrr Write register(s) ad address rrrrr
#define AD_OFSCAL  0x60  // Offset calibration
#define AD_GANCAL  0x61  // Gain calibration

// ADS1282 SPI
// max SCLK=fclk/2=4.096/2 MHz -> 2MHz
// SCLK polarity 0 (idles at 0)
// SCLK phase 0

#define SCLK  1000000

// delay of 24 fclk (6us) between bytes

#define SPI_DELAY   6

// ADS1282 Registers
// id      - reset value x0h ID3,ID2,ID1,ID0,0,0,0,0
// config0 - reset value 52h SYNC,1,DR2,DR1,DR0,PHS,FILTR1,FILTR0
// config1 - reset value 08h 0,MUX2,MUX1,MUX0,CHOP,PGA2,PGA1,PGA0
// hpf0    - reset value 32h HPF07,HPF06,HPF05,HPF04,HPF03,HPF02,HPF01,HPF00
// hpf1    - reset value 03h HPF15,HPF14,HPF13,HPF12,HPF11,HPF10,HPF09,HPF08
// ofc0    - reset value 00h OFC07,OFC06,OFC05,OFC04,OFC03,OFC02,OFC01,OFC00
// ofc1    - reset value 00h OFC15,OFC14,OFC13,OFC12,OFC11,OFC10,OFC09,OFC08
// ofc2    - reset value 00h OFC23,OFC22,OFC21,OFC20,OFC19,OFC18,OFC17,OFC16
// fsc0    - reset value 00h FSC07,FSC06,FSC05,FSC04,FSC03,FSC02,FSC01,FSC00
// fsc1    - reset value 00h FSC15,FSC14,FSC13,FSC12,FSC11,FSC10,FSC09,FSC08
// fsc2    - reset value 40h FSC23,FSC22,FSC21,FSC20,FSC19,FSC18,FSC17,FSC16

void SPI_RegsRead(SPI_HandleTypeDef *hspi, uint8_t *Registers);
void SPI_RegWrite(SPI_HandleTypeDef *hspi, int regnum, int regval);
void SPI_GetSample();

#endif /* SPI_H_ */
```

A.11 stm32h7xx_hal_msp.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * File Name          : stm32h7xx_hal_msp.c
  * Description        : This file provides code for the MSP Initialization
  *                      and de-Initialization codes.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */

/* USER CODE END Define */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */

/* USER CODE END Macro */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */

/* USER CODE END ExternalFunctions */
```

```
57
58  /* USER CODE BEGIN 0 */
59
60  /* USER CODE END 0 */
61  /**
62    * Initializes the Global MSP.
63    */
64  void HAL_MspInit(void)
65  {
66    /* USER CODE BEGIN MspInit 0 */
67
68    /* USER CODE END MspInit 0 */
69
70    __HAL_RCC_SYSCFG_CLK_ENABLE();
71
72    /* System interrupt init*/
73
74    /* USER CODE BEGIN MspInit 1 */
75
76    /* USER CODE END MspInit 1 */
77  }
78
79  /**
80  * @brief DAC MSP Initialization
81  * This function configures the hardware resources used in this example
82  * @param hdac: DAC handle pointer
83  * @retval None
84  */
85  void HAL_DAC_MspInit(DAC_HandleTypeDef* hdac)
86  {
87    GPIO_InitTypeDef GPIO_InitStruct = {0};
88    if(hdac->Instance==DAC1)
89    {
90    /* USER CODE BEGIN DAC1_MspInit 0 */
91
92    /* USER CODE END DAC1_MspInit 0 */
93      /* Peripheral clock enable */
94      __HAL_RCC_DAC12_CLK_ENABLE();
95
96      __HAL_RCC_GPIOA_CLK_ENABLE();
97      /**DAC1 GPIO Configuration
98      PA4     ------> DAC1_OUT1
99      */
100     GPIO_InitStruct.Pin = GPIO_PIN_4;
101     GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
102     GPIO_InitStruct.Pull = GPIO_NOPULL;
103     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
104
105   /* USER CODE BEGIN DAC1_MspInit 1 */
106
107   /* USER CODE END DAC1_MspInit 1 */
108   }
109
110 }
111
112 /**
113 * @brief DAC MSP De-Initialization
114 * This function freeze the hardware resources used in this example
115 * @param hdac: DAC handle pointer
```

```
116  * @retval None
117  */
118  void HAL_DAC_MspDeInit(DAC_HandleTypeDef* hdac)
119  {
120    if(hdac->Instance==DAC1)
121    {
122    /* USER CODE BEGIN DAC1_MspDeInit 0 */
123
124    /* USER CODE END DAC1_MspDeInit 0 */
125      /* Peripheral clock disable */
126      __HAL_RCC_DAC12_CLK_DISABLE();
127
128      /**DAC1 GPIO Configuration
129      PA4     ------> DAC1_OUT1
130      */
131      HAL_GPIO_DeInit(GPIOA, GPIO_PIN_4);
132
133    /* USER CODE BEGIN DAC1_MspDeInit 1 */
134
135    /* USER CODE END DAC1_MspDeInit 1 */
136    }
137
138  }
139
140  /**
141  * @brief SPI MSP Initialization
142  * This function configures the hardware resources used in this example
143  * @param hspi: SPI handle pointer
144  * @retval None
145  */
146  void HAL_SPI_MspInit(SPI_HandleTypeDef* hspi)
147  {
148    GPIO_InitTypeDef GPIO_InitStruct = {0};
149    if(hspi->Instance==SPI1)
150    {
151    /* USER CODE BEGIN SPI1_MspInit 0 */
152
153    /* USER CODE END SPI1_MspInit 0 */
154      /* Peripheral clock enable */
155      __HAL_RCC_SPI1_CLK_ENABLE();
156
157      __HAL_RCC_GPIOA_CLK_ENABLE();
158      __HAL_RCC_GPIOD_CLK_ENABLE();
159      /**SPI1 GPIO Configuration
160      PA5     ------> SPI1_SCK
161      PA6     ------> SPI1_MISO
162      PD7     ------> SPI1_MOSI
163      */
164      GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_6;
165      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
166      GPIO_InitStruct.Pull = GPIO_NOPULL;
167      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
168      GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
169      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
170
171      GPIO_InitStruct.Pin = GPIO_PIN_7;
172      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
173      GPIO_InitStruct.Pull = GPIO_NOPULL;
174      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
```

```
175       GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
176       HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
177
178     /* USER CODE BEGIN SPI1_MspInit 1 */
179
180     /* USER CODE END SPI1_MspInit 1 */
181     }
182     else if(hspi->Instance==SPI2)
183     {
184     /* USER CODE BEGIN SPI2_MspInit 0 */
185
186     /* USER CODE END SPI2_MspInit 0 */
187       /* Peripheral clock enable */
188       __HAL_RCC_SPI2_CLK_ENABLE();
189
190       __HAL_RCC_GPIOC_CLK_ENABLE();
191       __HAL_RCC_GPIOB_CLK_ENABLE();
192       /**SPI2 GPIO Configuration
193       PC2_C      ------> SPI2_MISO
194       PC3_C      ------> SPI2_MOSI
195       PB10       ------> SPI2_SCK
196       */
197       GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;
198       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
199       GPIO_InitStruct.Pull = GPIO_NOPULL;
200       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
201       GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
202       HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
203
204       GPIO_InitStruct.Pin = GPIO_PIN_10;
205       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
206       GPIO_InitStruct.Pull = GPIO_NOPULL;
207       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
208       GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
209       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
210
211     /* USER CODE BEGIN SPI2_MspInit 1 */
212
213     /* USER CODE END SPI2_MspInit 1 */
214     }
215     else if(hspi->Instance==SPI3)
216     {
217     /* USER CODE BEGIN SPI3_MspInit 0 */
218
219     /* USER CODE END SPI3_MspInit 0 */
220       /* Peripheral clock enable */
221       __HAL_RCC_SPI3_CLK_ENABLE();
222
223       __HAL_RCC_GPIOC_CLK_ENABLE();
224       /**SPI3 GPIO Configuration
225       PC10      ------> SPI3_SCK
226       PC11      ------> SPI3_MISO
227       PC12      ------> SPI3_MOSI
228       */
229       GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12;
230       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
231       GPIO_InitStruct.Pull = GPIO_NOPULL;
232       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
233       GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
```

```
234       HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
235
236     /* USER CODE BEGIN SPI3_MspInit 1 */
237
238     /* USER CODE END SPI3_MspInit 1 */
239     }
240     else if(hspi->Instance==SPI4)
241     {
242     /* USER CODE BEGIN SPI4_MspInit 0 */
243
244     /* USER CODE END SPI4_MspInit 0 */
245       /* Peripheral clock enable */
246       __HAL_RCC_SPI4_CLK_ENABLE();
247
248       __HAL_RCC_GPIOE_CLK_ENABLE();
249       /**SPI4 GPIO Configuration
250       PE2     ------> SPI4_SCK
251       PE5     ------> SPI4_MISO
252       PE6     ------> SPI4_MOSI
253       */
254       GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_5|GPIO_PIN_6;
255       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
256       GPIO_InitStruct.Pull = GPIO_NOPULL;
257       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
258       GPIO_InitStruct.Alternate = GPIO_AF5_SPI4;
259       HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
260
261     /* USER CODE BEGIN SPI4_MspInit 1 */
262
263     /* USER CODE END SPI4_MspInit 1 */
264     }
265     else if(hspi->Instance==SPI5)
266     {
267     /* USER CODE BEGIN SPI5_MspInit 0 */
268
269     /* USER CODE END SPI5_MspInit 0 */
270       /* Peripheral clock enable */
271       __HAL_RCC_SPI5_CLK_ENABLE();
272
273       __HAL_RCC_GPIOF_CLK_ENABLE();
274       /**SPI5 GPIO Configuration
275       PF7     ------> SPI5_SCK
276       PF8     ------> SPI5_MISO
277       PF9     ------> SPI5_MOSI
278       */
279       GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9;
280       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
281       GPIO_InitStruct.Pull = GPIO_NOPULL;
282       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
283       GPIO_InitStruct.Alternate = GPIO_AF5_SPI5;
284       HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
285
286     /* USER CODE BEGIN SPI5_MspInit 1 */
287
288     /* USER CODE END SPI5_MspInit 1 */
289     }
290     else if(hspi->Instance==SPI6)
291     {
292     /* USER CODE BEGIN SPI6_MspInit 0 */
```

```
293
294   /* USER CODE END SPI6_MspInit 0 */
295     /* Peripheral clock enable */
296     __HAL_RCC_SPI6_CLK_ENABLE();
297
298     __HAL_RCC_GPIOG_CLK_ENABLE();
299     __HAL_RCC_GPIOB_CLK_ENABLE();
300     /**SPI6 GPIO Configuration
301     PG12     ------> SPI6_MISO
302     PG14     ------> SPI6_MOSI
303     PB3 (JTDO/TRACESWO)     ------> SPI6_SCK
304     */
305     GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_14;
306     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
307     GPIO_InitStruct.Pull = GPIO_NOPULL;
308     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
309     GPIO_InitStruct.Alternate = GPIO_AF5_SPI6;
310     HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
311
312     GPIO_InitStruct.Pin = GPIO_PIN_3;
313     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
314     GPIO_InitStruct.Pull = GPIO_NOPULL;
315     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
316     GPIO_InitStruct.Alternate = GPIO_AF8_SPI6;
317     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
318
319   /* USER CODE BEGIN SPI6_MspInit 1 */
320
321   /* USER CODE END SPI6_MspInit 1 */
322   }
323
324 }
325
326 /**
327 * @brief SPI MSP De-Initialization
328 * This function freeze the hardware resources used in this example
329 * @param hspi: SPI handle pointer
330 * @retval None
331 */
332 void HAL_SPI_MspDeInit(SPI_HandleTypeDef* hspi)
333 {
334   if(hspi->Instance==SPI1)
335   {
336   /* USER CODE BEGIN SPI1_MspDeInit 0 */
337
338   /* USER CODE END SPI1_MspDeInit 0 */
339     /* Peripheral clock disable */
340     __HAL_RCC_SPI1_CLK_DISABLE();
341
342     /**SPI1 GPIO Configuration
343     PA5     ------> SPI1_SCK
344     PA6     ------> SPI1_MISO
345     PD7     ------> SPI1_MOSI
346     */
347     HAL_GPIO_DeInit(GPIOA, GPIO_PIN_5|GPIO_PIN_6);
348
349     HAL_GPIO_DeInit(GPIOD, GPIO_PIN_7);
350
351   /* USER CODE BEGIN SPI1_MspDeInit 1 */
```

```
352
353     /* USER CODE END SPI1_MspDeInit 1 */
354     }
355     else if(hspi->Instance==SPI2)
356     {
357     /* USER CODE BEGIN SPI2_MspDeInit 0 */
358
359     /* USER CODE END SPI2_MspDeInit 0 */
360       /* Peripheral clock disable */
361       __HAL_RCC_SPI2_CLK_DISABLE();
362
363       /**SPI2 GPIO Configuration
364       PC2_C     ------> SPI2_MISO
365       PC3_C     ------> SPI2_MOSI
366       PB10      ------> SPI2_SCK
367       */
368       HAL_GPIO_DeInit(GPIOC, GPIO_PIN_2|GPIO_PIN_3);
369
370       HAL_GPIO_DeInit(GPIOB, GPIO_PIN_10);
371
372     /* USER CODE BEGIN SPI2_MspDeInit 1 */
373
374     /* USER CODE END SPI2_MspDeInit 1 */
375     }
376     else if(hspi->Instance==SPI3)
377     {
378     /* USER CODE BEGIN SPI3_MspDeInit 0 */
379
380     /* USER CODE END SPI3_MspDeInit 0 */
381       /* Peripheral clock disable */
382       __HAL_RCC_SPI3_CLK_DISABLE();
383
384       /**SPI3 GPIO Configuration
385       PC10      ------> SPI3_SCK
386       PC11      ------> SPI3_MISO
387       PC12      ------> SPI3_MOSI
388       */
389       HAL_GPIO_DeInit(GPIOC, GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12);
390
391     /* USER CODE BEGIN SPI3_MspDeInit 1 */
392
393     /* USER CODE END SPI3_MspDeInit 1 */
394     }
395     else if(hspi->Instance==SPI4)
396     {
397     /* USER CODE BEGIN SPI4_MspDeInit 0 */
398
399     /* USER CODE END SPI4_MspDeInit 0 */
400       /* Peripheral clock disable */
401       __HAL_RCC_SPI4_CLK_DISABLE();
402
403       /**SPI4 GPIO Configuration
404       PE2      ------> SPI4_SCK
405       PE5      ------> SPI4_MISO
406       PE6      ------> SPI4_MOSI
407       */
408       HAL_GPIO_DeInit(GPIOE, GPIO_PIN_2|GPIO_PIN_5|GPIO_PIN_6);
409
410     /* USER CODE BEGIN SPI4_MspDeInit 1 */
```

```
411
412   /* USER CODE END SPI4_MspDeInit 1 */
413   }
414   else if(hspi->Instance==SPI5)
415   {
416   /* USER CODE BEGIN SPI5_MspDeInit 0 */
417
418   /* USER CODE END SPI5_MspDeInit 0 */
419     /* Peripheral clock disable */
420     __HAL_RCC_SPI5_CLK_DISABLE();
421
422     /**SPI5 GPIO Configuration
423     PF7     ------> SPI5_SCK
424     PF8     ------> SPI5_MISO
425     PF9     ------> SPI5_MOSI
426     */
427     HAL_GPIO_DeInit(GPIOF, GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9);
428
429   /* USER CODE BEGIN SPI5_MspDeInit 1 */
430
431   /* USER CODE END SPI5_MspDeInit 1 */
432   }
433   else if(hspi->Instance==SPI6)
434   {
435   /* USER CODE BEGIN SPI6_MspDeInit 0 */
436
437   /* USER CODE END SPI6_MspDeInit 0 */
438     /* Peripheral clock disable */
439     __HAL_RCC_SPI6_CLK_DISABLE();
440
441     /**SPI6 GPIO Configuration
442     PG12     ------> SPI6_MISO
443     PG14     ------> SPI6_MOSI
444     PB3 (JTDO/TRACESWO)     ------> SPI6_SCK
445     */
446     HAL_GPIO_DeInit(GPIOG, GPIO_PIN_12|GPIO_PIN_14);
447
448     HAL_GPIO_DeInit(GPIOB, GPIO_PIN_3);
449
450   /* USER CODE BEGIN SPI6_MspDeInit 1 */
451
452   /* USER CODE END SPI6_MspDeInit 1 */
453   }
454
455 }
456
457 /**
458 * @brief TIM_Base MSP Initialization
459 * This function configures the hardware resources used in this example
460 * @param htim_base: TIM_Base handle pointer
461 * @retval None
462 */
463 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base)
464 {
465   if(htim_base->Instance==TIM2)
466   {
467   /* USER CODE BEGIN TIM2_MspInit 0 */
468
469   /* USER CODE END TIM2_MspInit 0 */
```

```
470       /* Peripheral clock enable */
471       __HAL_RCC_TIM2_CLK_ENABLE();
472     /* USER CODE BEGIN TIM2_MspInit 1 */
473
474     /* USER CODE END TIM2_MspInit 1 */
475     }
476     else if(htim_base->Instance==TIM3)
477     {
478     /* USER CODE BEGIN TIM3_MspInit 0 */
479
480     /* USER CODE END TIM3_MspInit 0 */
481       /* Peripheral clock enable */
482       __HAL_RCC_TIM3_CLK_ENABLE();
483       /* TIM3 interrupt Init */
484       HAL_NVIC_SetPriority(TIM3_IRQn, 0, 0);
485       HAL_NVIC_EnableIRQ(TIM3_IRQn);
486     /* USER CODE BEGIN TIM3_MspInit 1 */
487
488     /* USER CODE END TIM3_MspInit 1 */
489     }
490
491   }
492
493   /**
494   * @brief TIM_Base MSP De-Initialization
495   * This function freeze the hardware resources used in this example
496   * @param htim_base: TIM_Base handle pointer
497   * @retval None
498   */
499   void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* htim_base)
500   {
501     if(htim_base->Instance==TIM2)
502     {
503     /* USER CODE BEGIN TIM2_MspDeInit 0 */
504
505     /* USER CODE END TIM2_MspDeInit 0 */
506       /* Peripheral clock disable */
507       __HAL_RCC_TIM2_CLK_DISABLE();
508     /* USER CODE BEGIN TIM2_MspDeInit 1 */
509
510     /* USER CODE END TIM2_MspDeInit 1 */
511     }
512     else if(htim_base->Instance==TIM3)
513     {
514     /* USER CODE BEGIN TIM3_MspDeInit 0 */
515
516     /* USER CODE END TIM3_MspDeInit 0 */
517       /* Peripheral clock disable */
518       __HAL_RCC_TIM3_CLK_DISABLE();
519
520       /* TIM3 interrupt DeInit */
521       HAL_NVIC_DisableIRQ(TIM3_IRQn);
522     /* USER CODE BEGIN TIM3_MspDeInit 1 */
523
524     /* USER CODE END TIM3_MspDeInit 1 */
525     }
526
527   }
528
```

```
529 /**
530 * @brief UART MSP Initialization
531 * This function configures the hardware resources used in this example
532 * @param huart: UART handle pointer
533 * @retval None
534 */
535 void HAL_UART_MspInit(UART_HandleTypeDef* huart)
536 {
537   GPIO_InitTypeDef GPIO_InitStruct = {0};
538   if(huart->Instance==USART3)
539   {
540   /* USER CODE BEGIN USART3_MspInit 0 */
541
542   /* USER CODE END USART3_MspInit 0 */
543     /* Peripheral clock enable */
544     __HAL_RCC_USART3_CLK_ENABLE();
545
546     __HAL_RCC_GPIOD_CLK_ENABLE();
547     /**USART3 GPIO Configuration
548     PD8     ------> USART3_TX
549     PD9     ------> USART3_RX
550     */
551     GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
552     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
553     GPIO_InitStruct.Pull = GPIO_NOPULL;
554     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
555     GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
556     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
557
558   /* USER CODE BEGIN USART3_MspInit 1 */
559
560   /* USER CODE END USART3_MspInit 1 */
561   }
562
563 }
564
565 /**
566 * @brief UART MSP De-Initialization
567 * This function freeze the hardware resources used in this example
568 * @param huart: UART handle pointer
569 * @retval None
570 */
571 void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
572 {
573   if(huart->Instance==USART3)
574   {
575   /* USER CODE BEGIN USART3_MspDeInit 0 */
576
577   /* USER CODE END USART3_MspDeInit 0 */
578     /* Peripheral clock disable */
579     __HAL_RCC_USART3_CLK_DISABLE();
580
581     /**USART3 GPIO Configuration
582     PD8     ------> USART3_TX
583     PD9     ------> USART3_RX
584     */
585     HAL_GPIO_DeInit(GPIOD, GPIO_PIN_8|GPIO_PIN_9);
586
587   /* USER CODE BEGIN USART3_MspDeInit 1 */
```

```
588
589     /* USER CODE END USART3_MspDeInit 1 */
590     }
591
592   }
593
594   /* USER CODE BEGIN 1 */
595
596   /* USER CODE END 1 */
597
598   /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.12 stm32h7xx_hal_timebase_tim.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32h7xx_hal_timebase_TIM.c
  * @brief   HAL time base based on the hardware TIM.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "stm32h7xx_hal.h"
#include "stm32h7xx_hal_tim.h"

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef        htim1;
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/

/**
  * @brief  This function configures the TIM1 as a time base source.
  *         The time source is configured  to have 1ms time base with a dedicated
  *         Tick interrupt priority.
  * @note   This function is called  automatically at the beginning of program after
  *         reset by HAL_Init() or at any time when clock is configured, by HAL_RCC_ClockConfig(
  * @param  TickPriority: Tick interrupt priority.
  * @retval HAL status
  */
HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
{
  RCC_ClkInitTypeDef    clkconfig;
  uint32_t              uwTimclock = 0;
  uint32_t              uwPrescalerValue = 0;
  uint32_t              pFLatency;

  /*Configure the TIM1 IRQ priority */
  HAL_NVIC_SetPriority(TIM1_UP_IRQn, TickPriority ,0);

  /* Enable the TIM1 global Interrupt */
  HAL_NVIC_EnableIRQ(TIM1_UP_IRQn);

  /* Enable TIM1 clock */
  __HAL_RCC_TIM1_CLK_ENABLE();
```

```
57
58    /* Get clock configuration */
59    HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
60
61    /* Compute TIM1 clock */
62    uwTimclock = 2*HAL_RCC_GetPCLK2Freq();
63
64    /* Compute the prescaler value to have TIM1 counter clock equal to 1MHz */
65    uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000) - 1);
66
67    /* Initialize TIM1 */
68    htim1.Instance = TIM1;
69
70    /* Initialize TIMx peripheral as follow:
71    + Period = [(TIM1CLK/1000) - 1]. to have a (1/1000) s time base.
72    + Prescaler = (uwTimclock/1000000 - 1) to have a 1MHz counter clock.
73    + ClockDivision = 0
74    + Counter direction = Up
75    */
76    htim1.Init.Period = (1000000 / 1000) - 1;
77    htim1.Init.Prescaler = uwPrescalerValue;
78    htim1.Init.ClockDivision = 0;
79    htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
80    if(HAL_TIM_Base_Init(&htim1) == HAL_OK)
81    {
82      /* Start the TIM time Base generation in interrupt mode */
83      return HAL_TIM_Base_Start_IT(&htim1);
84    }
85
86    /* Return function status */
87    return HAL_ERROR;
88  }
89
90  /**
91    * @brief  Suspend Tick increment.
92    * @note   Disable the tick increment by disabling TIM1 update interrupt.
93    * @param  None
94    * @retval None
95    */
96  void HAL_SuspendTick(void)
97  {
98    /* Disable TIM1 update Interrupt */
99    __HAL_TIM_DISABLE_IT(&htim1, TIM_IT_UPDATE);
100 }
101
102 /**
103   * @brief  Resume Tick increment.
104   * @note   Enable the tick increment by Enabling TIM1 update interrupt.
105   * @param  None
106   * @retval None
107   */
108 void HAL_ResumeTick(void)
109 {
110   /* Enable TIM1 Update interrupt */
111   __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_UPDATE);
112 }
113
114 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.13 stm32h7xx_hal_conf.h

```
/**
  ******************************************************************************
  * @file    stm32h7xx_hal_conf.h
  * @author  MCD Application Team
  * @brief   HAL configuration file.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32H7xx_HAL_CONF_H
#define __STM32H7xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED

  /* #define HAL_ADC_MODULE_ENABLED   */
/* #define HAL_FDCAN_MODULE_ENABLED   */
/* #define HAL_CEC_MODULE_ENABLED   */
/* #define HAL_COMP_MODULE_ENABLED   */
/* #define HAL_CRC_MODULE_ENABLED   */
/* #define HAL_CRYP_MODULE_ENABLED   */
#define HAL_DAC_MODULE_ENABLED
/* #define HAL_DCMI_MODULE_ENABLED   */
/* #define HAL_DMA2D_MODULE_ENABLED   */
#define HAL_ETH_MODULE_ENABLED
/* #define HAL_NAND_MODULE_ENABLED   */
/* #define HAL_NOR_MODULE_ENABLED   */
/* #define HAL_OTFDEC_MODULE_ENABLED   */
/* #define HAL_SRAM_MODULE_ENABLED   */
/* #define HAL_SDRAM_MODULE_ENABLED   */
/* #define HAL_HASH_MODULE_ENABLED   */
/* #define HAL_HRTIM_MODULE_ENABLED   */
/* #define HAL_HSEM_MODULE_ENABLED   */
/* #define HAL_GFXMMU_MODULE_ENABLED   */
/* #define HAL_JPEG_MODULE_ENABLED   */
```

```
57  /* #define HAL_OPAMP_MODULE_ENABLED   */
58  /* #define HAL_OSPI_MODULE_ENABLED   */
59  /* #define HAL_OSPI_MODULE_ENABLED   */
60  /* #define HAL_I2S_MODULE_ENABLED   */
61  /* #define HAL_SMBUS_MODULE_ENABLED   */
62  /* #define HAL_IWDG_MODULE_ENABLED   */
63  /* #define HAL_LPTIM_MODULE_ENABLED   */
64  /* #define HAL_LTDC_MODULE_ENABLED   */
65  /* #define HAL_QSPI_MODULE_ENABLED   */
66  /* #define HAL_RNG_MODULE_ENABLED   */
67  /* #define HAL_RTC_MODULE_ENABLED   */
68  /* #define HAL_SAI_MODULE_ENABLED   */
69  /* #define HAL_SD_MODULE_ENABLED   */
70  /* #define HAL_MMC_MODULE_ENABLED   */
71  /* #define HAL_SPDIFRX_MODULE_ENABLED   */
72  #define HAL_SPI_MODULE_ENABLED
73  /* #define HAL_SWPMI_MODULE_ENABLED   */
74  #define HAL_TIM_MODULE_ENABLED
75  #define HAL_UART_MODULE_ENABLED
76  /* #define HAL_USART_MODULE_ENABLED   */
77  /* #define HAL_IRDA_MODULE_ENABLED   */
78  /* #define HAL_SMARTCARD_MODULE_ENABLED   */
79  /* #define HAL_WWDG_MODULE_ENABLED   */
80  /* #define HAL_PCD_MODULE_ENABLED   */
81  /* #define HAL_HCD_MODULE_ENABLED   */
82  /* #define HAL_DFSDM_MODULE_ENABLED   */
83  /* #define HAL_DSI_MODULE_ENABLED   */
84  /* #define HAL_JPEG_MODULE_ENABLED   */
85  /* #define HAL_MDIOS_MODULE_ENABLED   */
86  /* #define HAL_PSSI_MODULE_ENABLED   */
87  /* #define HAL_DTS_MODULE_ENABLED   */
88  #define HAL_GPIO_MODULE_ENABLED
89  #define HAL_DMA_MODULE_ENABLED
90  #define HAL_MDMA_MODULE_ENABLED
91  #define HAL_RCC_MODULE_ENABLED
92  #define HAL_FLASH_MODULE_ENABLED
93  #define HAL_EXTI_MODULE_ENABLED
94  #define HAL_PWR_MODULE_ENABLED
95  #define HAL_I2C_MODULE_ENABLED
96  #define HAL_CORTEX_MODULE_ENABLED
97  #define HAL_HSEM_MODULE_ENABLED
98
99  /* ########################## Oscillator Values adaptation ####################*/
100 /**
101   * @brief Adjust the value of External High Speed oscillator (HSE) used in your application.
102   *        This value is used by the RCC HAL module to compute the system frequency
103   *        (when HSE is used as system clock source, directly or through the PLL).
104   */
105 #if !defined  (HSE_VALUE)
106 #define HSE_VALUE    ((uint32_t)8000000) /*!< Value of the External oscillator in Hz : FPGA case fixed
107 #endif /* HSE_VALUE */
108
109 #if !defined  (HSE_STARTUP_TIMEOUT)
110   #define HSE_STARTUP_TIMEOUT    ((uint32_t)100U)   /*!< Time out for HSE start up, in ms */
111 #endif /* HSE_STARTUP_TIMEOUT */
112
113 /**
114   * @brief Internal  oscillator (CSI) default value.
115   *        This value is the default CSI value after Reset.
```

```
116   */
117 #if !defined  (CSI_VALUE)
118   #define CSI_VALUE    ((uint32_t)4000000) /*!< Value of the Internal oscillator in Hz*/
119 #endif /* CSI_VALUE */
120
121 /**
122   * @brief Internal High Speed oscillator (HSI) value.
123   *        This value is used by the RCC HAL module to compute the system frequency
124   *        (when HSI is used as system clock source, directly or through the PLL).
125   */
126 #if !defined  (HSI_VALUE)
127   #define HSI_VALUE    ((uint32_t)64000000) /*!< Value of the Internal oscillator in Hz*/
128 #endif /* HSI_VALUE */
129
130 /**
131   * @brief External Low Speed oscillator (LSE) value.
132   *        This value is used by the UART, RTC HAL module to compute the system frequency
133   */
134 #if !defined  (LSE_VALUE)
135   #define LSE_VALUE    ((uint32_t)32768U) /*!< Value of the External oscillator in Hz*/
136 #endif /* LSE_VALUE */
137
138 #if !defined  (LSE_STARTUP_TIMEOUT)
139   #define LSE_STARTUP_TIMEOUT    ((uint32_t)5000U)   /*!< Time out for LSE start up, in ms */
140 #endif /* LSE_STARTUP_TIMEOUT */
141
142 /**
143   * @brief External clock source for I2S peripheral
144   *        This value is used by the I2S HAL module to compute the I2S clock source
145   *        frequency, this source is inserted directly through I2S_CKIN pad.
146   */
147 #if !defined  (EXTERNAL_CLOCK_VALUE)
148   #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of the External clock in Hz*/
149 #endif /* EXTERNAL_CLOCK_VALUE */
150
151 /* Tip: To avoid modifying this file each time you need to use different HSE,
152    ===  you can define the HSE value in your toolchain compiler preprocessor. */
153
154 /* ########################### System Configuration ######################### */
155 /**
156   * @brief This is the HAL system configuration section
157   */
158 #define  VDD_VALUE                    ((uint32_t)3300U) /*!< Value of VDD in mv */
159 #define  TICK_INT_PRIORITY            ((uint32_t)0U) /*!< tick interrupt priority */
160 #define  USE_RTOS                     0U
161 #define  USE_SD_TRANSCEIVER           0U                /*!< use uSD Transceiver */
162
163 #define  USE_HAL_ADC_REGISTER_CALLBACKS     0U /* ADC register callback disabled
    */
164 #define  USE_HAL_CEC_REGISTER_CALLBACKS     0U /* CEC register callback disabled
    */
165 #define  USE_HAL_COMP_REGISTER_CALLBACKS    0U /* COMP register callback disabled
    */
166 #define  USE_HAL_CRYP_REGISTER_CALLBACKS    0U /* CRYP register callback disabled
    */
167 #define  USE_HAL_DAC_REGISTER_CALLBACKS     0U /* DAC register callback disabled
    */
168 #define  USE_HAL_DCMI_REGISTER_CALLBACKS    0U /* DCMI register callback disabled
    */
```

```
169 #define  USE_HAL_DFSDM_REGISTER_CALLBACKS    0U /* DFSDM register callback disabled
    */
170 #define  USE_HAL_DMA2D_REGISTER_CALLBACKS    0U /* DMA2D register callback disabled
    */
171 #define  USE_HAL_DSI_REGISTER_CALLBACKS      0U /* DSI register callback disabled
    */
172 #define  USE_HAL_DTS_REGISTER_CALLBACKS      0U /* DTS register callback disabled
    */
173 #define  USE_HAL_ETH_REGISTER_CALLBACKS      0U /* ETH register callback disabled
    */
174 #define  USE_HAL_FDCAN_REGISTER_CALLBACKS    0U /* FDCAN register callback disabled
    */
175 #define  USE_HAL_NAND_REGISTER_CALLBACKS     0U /* NAND register callback disabled
    */
176 #define  USE_HAL_NOR_REGISTER_CALLBACKS      0U /* NOR register callback disabled
    */
177 #define  USE_HAL_SDRAM_REGISTER_CALLBACKS    0U /* SDRAM register callback disabled
    */
178 #define  USE_HAL_SRAM_REGISTER_CALLBACKS     0U /* SRAM register callback disabled
    */
179 #define  USE_HAL_HASH_REGISTER_CALLBACKS     0U /* HASH register callback disabled
    */
180 #define  USE_HAL_HCD_REGISTER_CALLBACKS      0U /* HCD register callback disabled
    */
181 #define  USE_HAL_GFXMMU_REGISTER_CALLBACKS   0U /* GFXMMU register callback disabled
    */
182 #define  USE_HAL_HRTIM_REGISTER_CALLBACKS    0U /* HRTIM register callback disabled
    */
183 #define  USE_HAL_I2C_REGISTER_CALLBACKS      0U /* I2C register callback disabled
    */
184 #define  USE_HAL_I2S_REGISTER_CALLBACKS      0U /* I2S register callback disabled
    */
185 #define  USE_HAL_IRDA_REGISTER_CALLBACKS     0U /* IRDA register callback disabled
    */
186 #define  USE_HAL_JPEG_REGISTER_CALLBACKS     0U /* JPEG register callback disabled
    */
187 #define  USE_HAL_LPTIM_REGISTER_CALLBACKS    0U /* LPTIM register callback disabled
    */
188 #define  USE_HAL_LTDC_REGISTER_CALLBACKS     0U /* LTDC register callback disabled
    */
189 #define  USE_HAL_MDIOS_REGISTER_CALLBACKS    0U /* MDIO register callback disabled
    */
190 #define  USE_HAL_MMC_REGISTER_CALLBACKS      0U /* MMC register callback disabled
    */
191 #define  USE_HAL_OPAMP_REGISTER_CALLBACKS    0U /* MDIO register callback disabled
    */
192 #define  USE_HAL_OSPI_REGISTER_CALLBACKS     0U /* OSPI register callback disabled
    */
193 #define  USE_HAL_OTFDEC_REGISTER_CALLBACKS   0U /* OTFDEC register callback disabled
    */
194 #define  USE_HAL_PCD_REGISTER_CALLBACKS      0U /* PCD register callback disabled
    */
195 #define  USE_HAL_QSPI_REGISTER_CALLBACKS     0U /* QSPI register callback disabled
    */
196 #define  USE_HAL_RNG_REGISTER_CALLBACKS      0U /* RNG register callback disabled
    */
197 #define  USE_HAL_RTC_REGISTER_CALLBACKS      0U /* RTC register callback disabled
    */
198 #define  USE_HAL_SAI_REGISTER_CALLBACKS      0U /* SAI register callback disabled
```

```
   */
199 #define  USE_HAL_SD_REGISTER_CALLBACKS      0U /* SD register callback disabled
   */
200 #define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS  0U /* SMARTCARD register callback disabled */
201 #define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */
202 #define  USE_HAL_SMBUS_REGISTER_CALLBACKS   0U /* SMBUS register callback disabled
   */
203 #define  USE_HAL_SPI_REGISTER_CALLBACKS     0U /* SPI register callback disabled
   */
204 #define  USE_HAL_SWPMI_REGISTER_CALLBACKS   0U /* SWPMI register callback disabled
   */
205 #define  USE_HAL_TIM_REGISTER_CALLBACKS     0U /* TIM register callback disabled
   */
206 #define  USE_HAL_UART_REGISTER_CALLBACKS    0U /* UART register callback disabled
   */
207 #define  USE_HAL_USART_REGISTER_CALLBACKS   0U /* USART register callback disabled
   */
208 #define  USE_HAL_WWDG_REGISTER_CALLBACKS    0U /* WWDG register callback disabled
   */
209
210 /* ########################## Ethernet Configuration ######################### */
211 #define ETH_TX_DESC_CNT         4  /* number of Ethernet Tx DMA descriptors */
212 #define ETH_RX_DESC_CNT         4  /* number of Ethernet Rx DMA descriptors */
213
214 #define ETH_MAC_ADDR0    ((uint8_t)0x02)
215 #define ETH_MAC_ADDR1    ((uint8_t)0x00)
216 #define ETH_MAC_ADDR2    ((uint8_t)0x00)
217 #define ETH_MAC_ADDR3    ((uint8_t)0x00)
218 #define ETH_MAC_ADDR4    ((uint8_t)0x00)
219 #define ETH_MAC_ADDR5    ((uint8_t)0x00)
220
221 /* ########################## Assert Selection ############################## */
222 /**
223   * @brief Uncomment the line below to expanse the "assert_param" macro in the
224   *        HAL drivers code
225   */
226 /* #define USE_FULL_ASSERT    1U */
227
228 /* Includes ------------------------------------------------------------------*/
229 /**
230   * @brief Include module's header file
231   */
232
233 #ifdef HAL_RCC_MODULE_ENABLED
234   #include "stm32h7xx_hal_rcc.h"
235 #endif /* HAL_RCC_MODULE_ENABLED */
236
237 #ifdef HAL_GPIO_MODULE_ENABLED
238   #include "stm32h7xx_hal_gpio.h"
239 #endif /* HAL_GPIO_MODULE_ENABLED */
240
241 #ifdef HAL_DMA_MODULE_ENABLED
242   #include "stm32h7xx_hal_dma.h"
243 #endif /* HAL_DMA_MODULE_ENABLED */
244
245 #ifdef HAL_MDMA_MODULE_ENABLED
246  #include "stm32h7xx_hal_mdma.h"
247 #endif /* HAL_MDMA_MODULE_ENABLED */
248
```

```
249 #ifdef HAL_HASH_MODULE_ENABLED
250   #include "stm32h7xx_hal_hash.h"
251 #endif /* HAL_HASH_MODULE_ENABLED */
252
253 #ifdef HAL_DCMI_MODULE_ENABLED
254   #include "stm32h7xx_hal_dcmi.h"
255 #endif /* HAL_DCMI_MODULE_ENABLED */
256
257 #ifdef HAL_DMA2D_MODULE_ENABLED
258   #include "stm32h7xx_hal_dma2d.h"
259 #endif /* HAL_DMA2D_MODULE_ENABLED */
260
261 #ifdef HAL_DSI_MODULE_ENABLED
262   #include "stm32h7xx_hal_dsi.h"
263 #endif /* HAL_DSI_MODULE_ENABLED */
264
265 #ifdef HAL_DFSDM_MODULE_ENABLED
266   #include "stm32h7xx_hal_dfsdm.h"
267 #endif /* HAL_DFSDM_MODULE_ENABLED */
268
269 #ifdef HAL_ETH_MODULE_ENABLED
270   #include "stm32h7xx_hal_eth.h"
271 #endif /* HAL_ETH_MODULE_ENABLED */
272
273 #ifdef HAL_EXTI_MODULE_ENABLED
274   #include "stm32h7xx_hal_exti.h"
275 #endif /* HAL_EXTI_MODULE_ENABLED */
276
277 #ifdef HAL_CORTEX_MODULE_ENABLED
278   #include "stm32h7xx_hal_cortex.h"
279 #endif /* HAL_CORTEX_MODULE_ENABLED */
280
281 #ifdef HAL_ADC_MODULE_ENABLED
282   #include "stm32h7xx_hal_adc.h"
283 #endif /* HAL_ADC_MODULE_ENABLED */
284
285 #ifdef HAL_FDCAN_MODULE_ENABLED
286   #include "stm32h7xx_hal_fdcan.h"
287 #endif /* HAL_FDCAN_MODULE_ENABLED */
288
289 #ifdef HAL_CEC_MODULE_ENABLED
290   #include "stm32h7xx_hal_cec.h"
291 #endif /* HAL_CEC_MODULE_ENABLED */
292
293 #ifdef HAL_COMP_MODULE_ENABLED
294   #include "stm32h7xx_hal_comp.h"
295 #endif /* HAL_COMP_MODULE_ENABLED */
296
297 #ifdef HAL_CRC_MODULE_ENABLED
298   #include "stm32h7xx_hal_crc.h"
299 #endif /* HAL_CRC_MODULE_ENABLED */
300
301 #ifdef HAL_CRYP_MODULE_ENABLED
302   #include "stm32h7xx_hal_cryp.h"
303 #endif /* HAL_CRYP_MODULE_ENABLED */
304
305 #ifdef HAL_DAC_MODULE_ENABLED
306   #include "stm32h7xx_hal_dac.h"
307 #endif /* HAL_DAC_MODULE_ENABLED */
```

```
308
309 #ifdef HAL_FLASH_MODULE_ENABLED
310   #include "stm32h7xx_hal_flash.h"
311 #endif /* HAL_FLASH_MODULE_ENABLED */
312
313 #ifdef HAL_GFXMMU_MODULE_ENABLED
314   #include "stm32h7xx_hal_gfxmmu.h"
315 #endif /* HAL_GFXMMU_MODULE_ENABLED */
316
317 #ifdef HAL_HRTIM_MODULE_ENABLED
318   #include "stm32h7xx_hal_hrtim.h"
319 #endif /* HAL_HRTIM_MODULE_ENABLED */
320
321 #ifdef HAL_HSEM_MODULE_ENABLED
322   #include "stm32h7xx_hal_hsem.h"
323 #endif /* HAL_HSEM_MODULE_ENABLED */
324
325 #ifdef HAL_SRAM_MODULE_ENABLED
326   #include "stm32h7xx_hal_sram.h"
327 #endif /* HAL_SRAM_MODULE_ENABLED */
328
329 #ifdef HAL_NOR_MODULE_ENABLED
330   #include "stm32h7xx_hal_nor.h"
331 #endif /* HAL_NOR_MODULE_ENABLED */
332
333 #ifdef HAL_NAND_MODULE_ENABLED
334   #include "stm32h7xx_hal_nand.h"
335 #endif /* HAL_NAND_MODULE_ENABLED */
336
337 #ifdef HAL_I2C_MODULE_ENABLED
338  #include "stm32h7xx_hal_i2c.h"
339 #endif /* HAL_I2C_MODULE_ENABLED */
340
341 #ifdef HAL_I2S_MODULE_ENABLED
342  #include "stm32h7xx_hal_i2s.h"
343 #endif /* HAL_I2S_MODULE_ENABLED */
344
345 #ifdef HAL_IWDG_MODULE_ENABLED
346  #include "stm32h7xx_hal_iwdg.h"
347 #endif /* HAL_IWDG_MODULE_ENABLED */
348
349 #ifdef HAL_JPEG_MODULE_ENABLED
350  #include "stm32h7xx_hal_jpeg.h"
351 #endif /* HAL_JPEG_MODULE_ENABLED */
352
353 #ifdef HAL_MDIOS_MODULE_ENABLED
354  #include "stm32h7xx_hal_mdios.h"
355 #endif /* HAL_MDIOS_MODULE_ENABLED */
356
357 #ifdef HAL_MMC_MODULE_ENABLED
358  #include "stm32h7xx_hal_mmc.h"
359 #endif /* HAL_MMC_MODULE_ENABLED */
360
361 #ifdef HAL_LPTIM_MODULE_ENABLED
362 #include "stm32h7xx_hal_lptim.h"
363 #endif /* HAL_LPTIM_MODULE_ENABLED */
364
365 #ifdef HAL_LTDC_MODULE_ENABLED
366 #include "stm32h7xx_hal_ltdc.h"
```

```
367 #endif /* HAL_LTDC_MODULE_ENABLED */
368
369 #ifdef HAL_OPAMP_MODULE_ENABLED
370 #include "stm32h7xx_hal_opamp.h"
371 #endif /* HAL_OPAMP_MODULE_ENABLED */
372
373 #ifdef HAL_OSPI_MODULE_ENABLED
374   #include "stm32h7xx_hal_ospi.h"
375 #endif /* HAL_OSPI_MODULE_ENABLED */
376
377 #ifdef HAL_OTFDEC_MODULE_ENABLED
378 #include "stm32h7xx_hal_otfdec.h"
379 #endif /* HAL_OTFDEC_MODULE_ENABLED */
380
381 #ifdef HAL_PWR_MODULE_ENABLED
382  #include "stm32h7xx_hal_pwr.h"
383 #endif /* HAL_PWR_MODULE_ENABLED */
384
385 #ifdef HAL_QSPI_MODULE_ENABLED
386  #include "stm32h7xx_hal_qspi.h"
387 #endif /* HAL_QSPI_MODULE_ENABLED */
388
389 #ifdef HAL_RAMECC_MODULE_ENABLED
390  #include "stm32h7xx_hal_ramecc.h"
391 #endif /* HAL_HCD_MODULE_ENABLED */
392
393 #ifdef HAL_RNG_MODULE_ENABLED
394  #include "stm32h7xx_hal_rng.h"
395 #endif /* HAL_RNG_MODULE_ENABLED */
396
397 #ifdef HAL_RTC_MODULE_ENABLED
398  #include "stm32h7xx_hal_rtc.h"
399 #endif /* HAL_RTC_MODULE_ENABLED */
400
401 #ifdef HAL_SAI_MODULE_ENABLED
402  #include "stm32h7xx_hal_sai.h"
403 #endif /* HAL_SAI_MODULE_ENABLED */
404
405 #ifdef HAL_SD_MODULE_ENABLED
406  #include "stm32h7xx_hal_sd.h"
407 #endif /* HAL_SD_MODULE_ENABLED */
408
409 #ifdef HAL_SDRAM_MODULE_ENABLED
410  #include "stm32h7xx_hal_sdram.h"
411 #endif /* HAL_SDRAM_MODULE_ENABLED */
412
413 #ifdef HAL_SPI_MODULE_ENABLED
414  #include "stm32h7xx_hal_spi.h"
415 #endif /* HAL_SPI_MODULE_ENABLED */
416
417 #ifdef HAL_SPDIFRX_MODULE_ENABLED
418  #include "stm32h7xx_hal_spdifrx.h"
419 #endif /* HAL_SPDIFRX_MODULE_ENABLED */
420
421 #ifdef HAL_SWPMI_MODULE_ENABLED
422  #include "stm32h7xx_hal_swpmi.h"
423 #endif /* HAL_SWPMI_MODULE_ENABLED */
424
425 #ifdef HAL_TIM_MODULE_ENABLED
```

```
426  #include "stm32h7xx_hal_tim.h"
427 #endif /* HAL_TIM_MODULE_ENABLED */
428
429 #ifdef HAL_UART_MODULE_ENABLED
430  #include "stm32h7xx_hal_uart.h"
431 #endif /* HAL_UART_MODULE_ENABLED */
432
433 #ifdef HAL_USART_MODULE_ENABLED
434  #include "stm32h7xx_hal_usart.h"
435 #endif /* HAL_USART_MODULE_ENABLED */
436
437 #ifdef HAL_IRDA_MODULE_ENABLED
438  #include "stm32h7xx_hal_irda.h"
439 #endif /* HAL_IRDA_MODULE_ENABLED */
440
441 #ifdef HAL_SMARTCARD_MODULE_ENABLED
442  #include "stm32h7xx_hal_smartcard.h"
443 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
444
445 #ifdef HAL_SMBUS_MODULE_ENABLED
446  #include "stm32h7xx_hal_smbus.h"
447 #endif /* HAL_SMBUS_MODULE_ENABLED */
448
449 #ifdef HAL_WWDG_MODULE_ENABLED
450  #include "stm32h7xx_hal_wwdg.h"
451 #endif /* HAL_WWDG_MODULE_ENABLED */
452
453 #ifdef HAL_PCD_MODULE_ENABLED
454  #include "stm32h7xx_hal_pcd.h"
455 #endif /* HAL_PCD_MODULE_ENABLED */
456
457 #ifdef HAL_HCD_MODULE_ENABLED
458  #include "stm32h7xx_hal_hcd.h"
459 #endif /* HAL_HCD_MODULE_ENABLED */
460
461 #ifdef HAL_PSSI_MODULE_ENABLED
462   #include "stm32h7xx_hal_pssi.h"
463 #endif /* HAL_PSSI_MODULE_ENABLED */
464
465 #ifdef HAL_DTS_MODULE_ENABLED
466   #include "stm32h7xx_hal_dts.h"
467 #endif /* HAL_DTS_MODULE_ENABLED */
468
469 /* Exported macro ------------------------------------------------------------*/
470 #ifdef  USE_FULL_ASSERT
471 /**
472   * @brief  The assert_param macro is used for function's parameters check.
473   * @param  expr: If expr is false, it calls assert_failed function
474   *         which reports the name of the source file and the source
475   *         line number of the call that failed.
476   *         If expr is true, it returns no value.
477   * @retval None
478   */
479   #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
480 /* Exported functions ------------------------------------------------------- */
481   void assert_failed(uint8_t* file, uint32_t line);
482 #else
483   #define assert_param(expr) ((void)0U)
484 #endif /* USE_FULL_ASSERT */
```

```
485
486 #ifdef __cplusplus
487 }
488 #endif
489
490 #endif /* __STM32H7xx_HAL_CONF_H */
491
492
493 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.14 stm32h7xx_it.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32h7xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32h7xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
```

```
57
58  /* External variables ---------------------------------------------------------*/
59  extern ETH_HandleTypeDef heth;
60  extern TIM_HandleTypeDef htim3;
61  extern TIM_HandleTypeDef htim1;
62
63  /* USER CODE BEGIN EV */
64
65  /* USER CODE END EV */
66
67  /******************************************************************************/
68  /*           Cortex Processor Interruption and Exception Handlers          */
69  /******************************************************************************/
70  /**
71    * @brief This function handles Non maskable interrupt.
72    */
73  void NMI_Handler(void)
74  {
75    /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
76
77    /* USER CODE END NonMaskableInt_IRQn 0 */
78    /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
79
80    /* USER CODE END NonMaskableInt_IRQn 1 */
81  }
82
83  /**
84    * @brief This function handles Hard fault interrupt.
85    */
86  void HardFault_Handler(void)
87  {
88    /* USER CODE BEGIN HardFault_IRQn 0 */
89
90    /* USER CODE END HardFault_IRQn 0 */
91    while (1)
92    {
93      /* USER CODE BEGIN W1_HardFault_IRQn 0 */
94      /* USER CODE END W1_HardFault_IRQn 0 */
95    }
96  }
97
98  /**
99    * @brief This function handles Memory management fault.
100   */
101 void MemManage_Handler(void)
102 {
103   /* USER CODE BEGIN MemoryManagement_IRQn 0 */
104
105   /* USER CODE END MemoryManagement_IRQn 0 */
106   while (1)
107   {
108     /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
109     /* USER CODE END W1_MemoryManagement_IRQn 0 */
110   }
111 }
112
113 /**
114   * @brief This function handles Pre-fetch fault, memory access fault.
115   */
```

```
116 void BusFault_Handler(void)
117 {
118   /* USER CODE BEGIN BusFault_IRQn 0 */
119
120   /* USER CODE END BusFault_IRQn 0 */
121   while (1)
122   {
123     /* USER CODE BEGIN W1_BusFault_IRQn 0 */
124     /* USER CODE END W1_BusFault_IRQn 0 */
125   }
126 }
127
128 /**
129   * @brief This function handles Undefined instruction or illegal state.
130   */
131 void UsageFault_Handler(void)
132 {
133   /* USER CODE BEGIN UsageFault_IRQn 0 */
134
135   /* USER CODE END UsageFault_IRQn 0 */
136   while (1)
137   {
138     /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
139     /* USER CODE END W1_UsageFault_IRQn 0 */
140   }
141 }
142
143 /**
144   * @brief This function handles System service call via SWI instruction.
145   */
146 void SVC_Handler(void)
147 {
148   /* USER CODE BEGIN SVCall_IRQn 0 */
149
150   /* USER CODE END SVCall_IRQn 0 */
151   /* USER CODE BEGIN SVCall_IRQn 1 */
152
153   /* USER CODE END SVCall_IRQn 1 */
154 }
155
156 /**
157   * @brief This function handles Debug monitor.
158   */
159 void DebugMon_Handler(void)
160 {
161   /* USER CODE BEGIN DebugMonitor_IRQn 0 */
162
163   /* USER CODE END DebugMonitor_IRQn 0 */
164   /* USER CODE BEGIN DebugMonitor_IRQn 1 */
165
166   /* USER CODE END DebugMonitor_IRQn 1 */
167 }
168
169 /**
170   * @brief This function handles Pendable request for system service.
171   */
172 void PendSV_Handler(void)
173 {
174   /* USER CODE BEGIN PendSV_IRQn 0 */
```

```
175
176   /* USER CODE END PendSV_IRQn 0 */
177   /* USER CODE BEGIN PendSV_IRQn 1 */
178
179   /* USER CODE END PendSV_IRQn 1 */
180 }
181
182 /******************************************************************************/
183 /* STM32H7xx Peripheral Interrupt Handlers                                    */
184 /* Add here the Interrupt Handlers for the used peripherals.                  */
185 /* For the available peripheral interrupt handler names,                      */
186 /* please refer to the startup file (startup_stm32h7xx.s).                    */
187 /******************************************************************************/
188
189 /**
190   * @brief This function handles EXTI line[9:5] interrupts.
191   */
192 void EXTI9_5_IRQHandler(void)
193 {
194   /* USER CODE BEGIN EXTI9_5_IRQn 0 */
195
196   /* USER CODE END EXTI9_5_IRQn 0 */
197   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_6);
198   /* USER CODE BEGIN EXTI9_5_IRQn 1 */
199
200   /* USER CODE END EXTI9_5_IRQn 1 */
201 }
202
203 /**
204   * @brief This function handles TIM1 update interrupt.
205   */
206 void TIM1_UP_IRQHandler(void)
207 {
208   /* USER CODE BEGIN TIM1_UP_IRQn 0 */
209
210   /* USER CODE END TIM1_UP_IRQn 0 */
211   HAL_TIM_IRQHandler(&htim1);
212   /* USER CODE BEGIN TIM1_UP_IRQn 1 */
213
214   /* USER CODE END TIM1_UP_IRQn 1 */
215 }
216
217 /**
218   * @brief This function handles TIM3 global interrupt.
219   */
220 void TIM3_IRQHandler(void)
221 {
222   /* USER CODE BEGIN TIM3_IRQn 0 */
223
224   /* USER CODE END TIM3_IRQn 0 */
225   HAL_TIM_IRQHandler(&htim3);
226   /* USER CODE BEGIN TIM3_IRQn 1 */
227
228   /* USER CODE END TIM3_IRQn 1 */
229 }
230
231 /**
232   * @brief This function handles EXTI line[15:10] interrupts.
233   */
```

```
234 void EXTI15_10_IRQHandler(void)
235 {
236   /* USER CODE BEGIN EXTI15_10_IRQn 0 */
237
238   /* USER CODE END EXTI15_10_IRQn 0 */
239   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_11);
240   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
241   /* USER CODE BEGIN EXTI15_10_IRQn 1 */
242
243   /* USER CODE END EXTI15_10_IRQn 1 */
244 }
245
246 /**
247   * @brief This function handles Ethernet global interrupt.
248   */
249 void ETH_IRQHandler(void)
250 {
251   /* USER CODE BEGIN ETH_IRQn 0 */
252
253   /* USER CODE END ETH_IRQn 0 */
254   HAL_ETH_IRQHandler(&heth);
255   /* USER CODE BEGIN ETH_IRQn 1 */
256
257   /* USER CODE END ETH_IRQn 1 */
258 }
259
260 /* USER CODE BEGIN 1 */
261
262 /* USER CODE END 1 */
263 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.15 stm32h7xx_it.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32h7xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
 ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32H7xx_IT_H
#define __STM32H7xx_IT_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
```

```
57 void PendSV_Handler(void);
58 void EXTI9_5_IRQHandler(void);
59 void TIM1_UP_IRQHandler(void);
60 void TIM3_IRQHandler(void);
61 void EXTI15_10_IRQHandler(void);
62 void ETH_IRQHandler(void);
63 /* USER CODE BEGIN EFP */
64
65 /* USER CODE END EFP */
66
67 #ifdef __cplusplus
68 }
69 #endif
70
71 #endif /* __STM32H7xx_IT_H */
72
73 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.16 syscalls.c

```
/**
******************************************************************************
**
**  File        : syscalls.c
**
**  Author      : Auto-generated by STM32CubeIDE
**
**  Abstract    : STM32CubeIDE Minimal System calls file
**
**                For more information about which c-functions
**                need which of these lowlevel functions
**                please consult the Newlib libc-manual
**
**  Environment : STM32CubeIDE MCU
**
**  Distribution: The file is distributed as is, without any warranty
**                of any kind.
**
******************************************************************************
**
** <h2><center>&copy; COPYRIGHT(c) 2018 STMicroelectronics</center></h2>
**
** Redistribution and use in source and binary forms, with or without modification,
** are permitted provided that the following conditions are met:
**   1. Redistributions of source code must retain the above copyright notice,
**      this list of conditions and the following disclaimer.
**   2. Redistributions in binary form must reproduce the above copyright notice,
**      this list of conditions and the following disclaimer in the documentation
**      and/or other materials provided with the distribution.
**   3. Neither the name of STMicroelectronics nor the names of its contributors
**      may be used to endorse or promote products derived from this software
**      without specific prior written permission.
**
** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
** AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
** IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
** DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
** FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
** DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
** SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
** CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
** OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
** OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**
**
******************************************************************************
*/

/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
```

```
57 #include <sys/times.h>
58
59
60 /* Variables */
61 //#undef errno
62 extern int errno;
63 extern int __io_putchar(int ch) __attribute__((weak));
64 extern int __io_getchar(void) __attribute__((weak));
65
66 register char * stack_ptr asm("sp");
67
68 char *__env[1] = { 0 };
69 char **environ = __env;
70
71
72 /* Functions */
73 void initialise_monitor_handles()
74 {
75 }
76
77 int _getpid(void)
78 {
79     return 1;
80 }
81
82 int _kill(int pid, int sig)
83 {
84     errno = EINVAL;
85     return -1;
86 }
87
88 void _exit (int status)
89 {
90     _kill(status, -1);
91     while (1) {}        /* Make sure we hang here */
92 }
93
94 __attribute__((weak)) int _read(int file, char *ptr, int len)
95 {
96     int DataIdx;
97
98     for (DataIdx = 0; DataIdx < len; DataIdx++)
99     {
100        *ptr++ = __io_getchar();
101    }
102
103 return len;
104 }
105
106 __attribute__((weak)) int _write(int file, char *ptr, int len)
107 {
108     int DataIdx;
109
110     for (DataIdx = 0; DataIdx < len; DataIdx++)
111     {
112         __io_putchar(*ptr++);
113     }
114     return len;
115 }
```

```
116
117 int _close(int file)
118 {
119     return -1;
120 }
121
122
123 int _fstat(int file, struct stat *st)
124 {
125     st->st_mode = S_IFCHR;
126     return 0;
127 }
128
129 int _isatty(int file)
130 {
131     return 1;
132 }
133
134 int _lseek(int file, int ptr, int dir)
135 {
136     return 0;
137 }
138
139 int _open(char *path, int flags, ...)
140 {
141     /* Pretend like we always fail */
142     return -1;
143 }
144
145 int _wait(int *status)
146 {
147     errno = ECHILD;
148     return -1;
149 }
150
151 int _unlink(char *name)
152 {
153     errno = ENOENT;
154     return -1;
155 }
156
157 int _times(struct tms *buf)
158 {
159     return -1;
160 }
161
162 int _stat(char *file, struct stat *st)
163 {
164     st->st_mode = S_IFCHR;
165     return 0;
166 }
167
168 int _link(char *old, char *new)
169 {
170     errno = EMLINK;
171     return -1;
172 }
173
174 int _fork(void)
```

```
175 {
176     errno = EAGAIN;
177     return -1;
178 }
179
180 int _execve(char *name, char **argv, char **env)
181 {
182     errno = ENOMEM;
183     return -1;
184 }
```

A.17 sysmem.c

```
/**
******************************************************************************
**
**  File        : sysmem.c
**
**  Author      : Auto-generated by STM32CubeIDE
**
**  Abstract    : STM32CubeIDE Minimal System Memory calls file
**
**                For more information about which c-functions
**                need which of these lowlevel functions
**                please consult the Newlib libc-manual
**
**  Environment : STM32CubeIDE MCU
**
**  Distribution: The file is distributed as is, without any warranty
**                of any kind.
**
******************************************************************************
**
** <h2><center>&copy; COPYRIGHT(c) 2018 STMicroelectronics</center></h2>
**
** Redistribution and use in source and binary forms, with or without modification,
** are permitted provided that the following conditions are met:
**   1. Redistributions of source code must retain the above copyright notice,
**      this list of conditions and the following disclaimer.
**   2. Redistributions in binary form must reproduce the above copyright notice,
**      this list of conditions and the following disclaimer in the documentation
**      and/or other materials provided with the distribution.
**   3. Neither the name of STMicroelectronics nor the names of its contributors
**      may be used to endorse or promote products derived from this software
**      without specific prior written permission.
**
** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
** AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
** IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
** DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
** FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
** DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
** SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
** CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
** OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
** OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**
**
******************************************************************************
*/

/* Includes */
#include <errno.h>
#include <stdio.h>

/* Variables */
extern int errno;
register char * stack_ptr asm("sp");

```

```
57 /* Functions */
58
59 /**
60  _sbrk
61  Increase program data space. Malloc and related functions depend on this
62 **/
63 caddr_t _sbrk(int incr)
64 {
65     extern char end asm("end");
66     static char *heap_end;
67     char *prev_heap_end;
68
69     if (heap_end == 0)
70         heap_end = &end;
71
72     prev_heap_end = heap_end;
73     if (heap_end + incr > stack_ptr)
74     {
75         errno = ENOMEM;
76         return (caddr_t) -1;
77     }
78
79     heap_end += incr;
80
81     return (caddr_t) prev_heap_end;
82 }
```

A.18 system_stm32h7xx.c

```
/**
  ******************************************************************************
  * @file    system_stm32h7xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-Mx Device Peripheral Access Layer System Source File.
  *
  *   This file provides two functions and one global variable to be called from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32h7xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it can be used
  *                                  by the user application to setup the SysTick
  *                                  timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must
  *                                 be called whenever the core clock is changed
  *                                 during program execution.
  *
  *
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */

/** @addtogroup CMSIS
  * @{
  */

/** @addtogroup stm32h7xx_system
  * @{
  */

/** @addtogroup STM32H7xx_System_Private_Includes
  * @{
  */

#include "stm32h7xx.h"
#include <math.h>
#if !defined  (HSE_VALUE)
#define HSE_VALUE    ((uint32_t)25000000) /*!< Value of the External oscillator in Hz */
#endif /* HSE_VALUE */

#if !defined  (CSI_VALUE)
  #define CSI_VALUE    ((uint32_t)4000000) /*!< Value of the Internal oscillator in Hz*/
#endif /* CSI_VALUE */
```

```
57
58 #if !defined  (HSI_VALUE)
59   #define HSI_VALUE    ((uint32_t)64000000) /*!< Value of the Internal oscillator in Hz*/
60 #endif /* HSI_VALUE */
61
62
63 /**
64   * @}
65   */
66
67 /** @addtogroup STM32H7xx_System_Private_TypesDefinitions
68   * @{
69   */
70
71 /**
72   * @}
73   */
74
75 /** @addtogroup STM32H7xx_System_Private_Defines
76   * @{
77   */
78
79 /************************* Miscellaneous Configuration ************************/
80 /*!< Uncomment the following line if you need to use initialized data in D2 domain SRAM (AHB SRAM
81 /* #define DATA_IN_D2_SRAM */
82
83 /*!< Uncomment the following line if you need to relocate your vector Table in
84      Internal SRAM. */
85 /* #define VECT_TAB_SRAM */
86 #define VECT_TAB_OFFSET  0x00000000UL /*!< Vector Table base offset field.
87                                        This value must be a multiple of 0x200. */
88 /******************************************************************************/
89
90 /**
91   * @}
92   */
93
94 /** @addtogroup STM32H7xx_System_Private_Macros
95   * @{
96   */
97
98 /**
99   * @}
100   */
101
102 /** @addtogroup STM32H7xx_System_Private_Variables
103   * @{
104   */
105   /* This variable is updated in three ways:
106       1) by calling CMSIS function SystemCoreClockUpdate()
107       2) by calling HAL API function HAL_RCC_GetHCLKFreq()
108       3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
109          Note: If you use this function to configure the system clock; then there
110                is no need to call the 2 first functions listed above, since SystemCoreClock
111                variable is updated automatically.
112   */
113   uint32_t SystemCoreClock = 64000000;
114   uint32_t SystemD2Clock = 64000000;
115   const  uint8_t D1CorePrescTable[16] = {0, 0, 0, 0, 1, 2, 3, 4, 1, 2, 3, 4, 6, 7, 8, 9};
```

```
116
117 /**
118   * @}
119   */
120
121 /** @addtogroup STM32H7xx_System_Private_FunctionPrototypes
122   * @{
123   */
124
125 /**
126   * @}
127   */
128
129 /** @addtogroup STM32H7xx_System_Private_Functions
130   * @{
131   */
132
133 /**
134   * @brief  Setup the microcontroller system
135   *         Initialize the FPU setting and  vector table location
136   *         configuration.
137   * @param  None
138   * @retval None
139   */
140 void SystemInit (void)
141 {
142 #if defined (DATA_IN_D2_SRAM)
143   __IO uint32_t tmpreg;
144 #endif /* DATA_IN_D2_SRAM */
145
146   /* FPU settings ------------------------------------------------------------*/
147   #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
148     SCB->CPACR |= ((3UL << (10*2))|(3UL << (11*2)));  /* set CP10 and CP11 Full Access */
149   #endif
150   /* Reset the RCC clock configuration to the default reset state ------------*/
151   /* Set HSION bit */
152   RCC->CR |= RCC_CR_HSION;
153
154   /* Reset CFGR register */
155   RCC->CFGR = 0x00000000;
156
157   /* Reset HSEON, CSSON , CSION,RC48ON, CSIKERON PLL1ON, PLL2ON and PLL3ON bits */
158   RCC->CR &= 0xEAF6ED7FU;
159
160   /* Reset D1CFGR register */
161   RCC->D1CFGR = 0x00000000;
162
163   /* Reset D2CFGR register */
164   RCC->D2CFGR = 0x00000000;
165
166   /* Reset D3CFGR register */
167   RCC->D3CFGR = 0x00000000;
168
169   /* Reset PLLCKSELR register */
170   RCC->PLLCKSELR = 0x00000000;
171
172   /* Reset PLLCFGR register */
173   RCC->PLLCFGR = 0x00000000;
174   /* Reset PLL1DIVR register */
```

```
175     RCC->PLL1DIVR = 0x00000000;
176     /* Reset PLL1FRACR register */
177     RCC->PLL1FRACR = 0x00000000;
178
179     /* Reset PLL2DIVR register */
180     RCC->PLL2DIVR = 0x00000000;
181
182     /* Reset PLL2FRACR register */
183
184     RCC->PLL2FRACR = 0x00000000;
185     /* Reset PLL3DIVR register */
186     RCC->PLL3DIVR = 0x00000000;
187
188     /* Reset PLL3FRACR register */
189     RCC->PLL3FRACR = 0x00000000;
190
191     /* Reset HSEBYP bit */
192     RCC->CR &= 0xFFFBFFFFU;
193
194     /* Disable all interrupts */
195     RCC->CIER = 0x00000000;
196
197 #if defined (DATA_IN_D2_SRAM)
198     /* in case of initialized data in D2 SRAM (AHB SRAM) , enable the D2 SRAM clock ((AHB SRAM clo
199 #if defined(RCC_AHB2ENR_D2SRAM1EN)
200     RCC->AHB2ENR |= (RCC_AHB2ENR_D2SRAM1EN | RCC_AHB2ENR_D2SRAM2EN | RCC_AHB2ENR_D2SRAM3EN);
201 #else
202     RCC->AHB2ENR |= (RCC_AHB2ENR_AHBSRAM1EN | RCC_AHB2ENR_AHBSRAM2EN);
203 #endif /* RCC_AHB2ENR_D2SRAM1EN */
204
205     tmpreg = RCC->AHB2ENR;
206     (void) tmpreg;
207 #endif /* DATA_IN_D2_SRAM */
208
209 #if defined(DUAL_CORE) && defined(CORE_CM4)
210     /* Configure the Vector Table location add offset address for cortex-M4 ------------------*/
211 #ifdef VECT_TAB_SRAM
212     SCB->VTOR = D2_AHBSRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
213 #else
214     SCB->VTOR = FLASH_BANK2_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH *
215 #endif /* VECT_TAB_SRAM */
216
217 #else
218     /* dual core CM7 or single core line */
219     if((DBGMCU->IDCODE & 0xFFFF0000U) < 0x20000000U)
220     {
221       /* if stm32h7 revY*/
222       /* Change  the switch matrix read issuing capability to 1 for the AXI SRAM target (Target 7)
223       *((__IO uint32_t*)0x51008108) = 0x000000001U;
224     }
225
226     /* Configure the Vector Table location add offset address for cortex-M7 ------------------*/
227 #ifdef VECT_TAB_SRAM
228     SCB->VTOR = D1_AXISRAM_BASE  | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal D1 AXI-
229 #else
230     SCB->VTOR = FLASH_BANK1_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH *
231 #endif
232
233 #endif /*DUAL_CORE && CORE_CM4*/
```

```
234
235 }
236
237 /**
238   * @brief  Update SystemCoreClock variable according to Clock Register Values.
239   *         The SystemCoreClock variable contains the core clock , it can
240   *         be used by the user application to setup the SysTick timer or configure
241   *         other parameters.
242   *
243   * @note   Each time the core clock changes, this function must be called
244   *         to update SystemCoreClock variable value. Otherwise, any configuration
245   *         based on this variable will be incorrect.
246   *
247   * @note   - The system frequency computed by this function is not the real
248   *           frequency in the chip. It is calculated based on the predefined
249   *           constant and the selected clock source:
250   *
251   *           - If SYSCLK source is CSI, SystemCoreClock will contain the CSI_VALUE(*)
252   *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(**)
253   *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(***)
254   *           - If SYSCLK source is PLL, SystemCoreClock will contain the CSI_VALUE(*),
255   *             HSI_VALUE(**) or HSE_VALUE(***) multiplied/divided by the PLL factors.
256   *
257   *         (*) CSI_VALUE is a constant defined in stm32h7xx_hal.h file (default value
258   *             4 MHz) but the real value may vary depending on the variations
259   *             in voltage and temperature.
260   *         (**) HSI_VALUE is a constant defined in stm32h7xx_hal.h file (default value
261   *             64 MHz) but the real value may vary depending on the variations
262   *             in voltage and temperature.
263   *
264   *         (***)HSE_VALUE is a constant defined in stm32h7xx_hal.h file (default value
265   *              25 MHz), user has to ensure that HSE_VALUE is same as the real
266   *              frequency of the crystal used. Otherwise, this function may
267   *              have wrong result.
268   *
269   *         - The result of this function could be not correct when using fractional
270   *           value for HSE crystal.
271   * @param  None
272   * @retval None
273   */
274 void SystemCoreClockUpdate (void)
275 {
276   uint32_t pllp, pllsource, pllm, pllfracen, hsivalue, tmp;
277   float_t fracn1, pllvco;
278
279   /* Get SYSCLK source -------------------------------------------------------*/
280
281   switch (RCC->CFGR & RCC_CFGR_SWS)
282   {
283   case RCC_CFGR_SWS_HSI:  /* HSI used as system clock source */
284     SystemCoreClock = (uint32_t) (HSI_VALUE >> ((RCC->CR & RCC_CR_HSIDIV)>> 3));
285     break;
286
287   case RCC_CFGR_SWS_CSI:  /* CSI used as system clock  source */
288     SystemCoreClock = CSI_VALUE;
289     break;
290
291   case RCC_CFGR_SWS_HSE:  /* HSE used as system clock  source */
292     SystemCoreClock = HSE_VALUE;
```

```
293       break;
294
295     case RCC_CFGR_SWS_PLL1:  /* PLL1 used as system clock  source */
296
297       /* PLL_VCO = (HSE_VALUE or HSI_VALUE or CSI_VALUE/ PLLM) * PLLN
298       SYSCLK = PLL_VCO / PLLR
299       */
300       pllsource = (RCC->PLLCKSELR & RCC_PLLCKSELR_PLLSRC);
301       pllm = ((RCC->PLLCKSELR & RCC_PLLCKSELR_DIVM1)>> 4)  ;
302       pllfracen = ((RCC->PLLCFGR & RCC_PLLCFGR_PLL1FRACEN)>>RCC_PLLCFGR_PLL1FRACEN_Pos);
303       fracn1 = (float_t)(uint32_t)(pllfracen* ((RCC->PLL1FRACR & RCC_PLL1FRACR_FRACN1)>> 3));
304
305       if (pllm != 0U)
306       {
307         switch (pllsource)
308         {
309           case RCC_PLLCKSELR_PLLSRC_HSI:  /* HSI used as PLL clock source */
310
311           hsivalue = (HSI_VALUE >> ((RCC->CR & RCC_CR_HSIDIV)>> 3)) ;
312           pllvco = ( (float_t)hsivalue / (float_t)pllm) * ((float_t)(uint32_t)(RCC->PLL1DIVR & RCC
313
314           break;
315
316           case RCC_PLLCKSELR_PLLSRC_CSI:  /* CSI used as PLL clock source */
317             pllvco = ((float_t)CSI_VALUE / (float_t)pllm) * ((float_t)(uint32_t)(RCC->PLL1DIVR & R
318           break;
319
320           case RCC_PLLCKSELR_PLLSRC_HSE:  /* HSE used as PLL clock source */
321             pllvco = ((float_t)HSE_VALUE / (float_t)pllm) * ((float_t)(uint32_t)(RCC->PLL1DIVR & R
322           break;
323
324         default:
325             pllvco = ((float_t)CSI_VALUE / (float_t)pllm) * ((float_t)(uint32_t)(RCC->PLL1DIVR & R
326           break;
327         }
328         pllp = (((RCC->PLL1DIVR & RCC_PLL1DIVR_P1) >>9) + 1U ) ;
329         SystemCoreClock =  (uint32_t)(float_t)(pllvco/(float_t)pllp);
330       }
331       else
332       {
333         SystemCoreClock = 0U;
334       }
335       break;
336
337     default:
338       SystemCoreClock = CSI_VALUE;
339       break;
340     }
341
342     /* Compute SystemClock frequency ---------------------------------------------------*/
343     tmp = D1CorePrescTable[(RCC->D1CFGR & RCC_D1CFGR_D1CPRE)>> RCC_D1CFGR_D1CPRE_Pos];
344
345     /* SystemCoreClock frequency : CM7 CPU frequency  */
346     SystemCoreClock >>= tmp;
347
348     /* SystemD2Clock frequency : CM4 CPU, AXI and AHBs Clock frequency  */
349     SystemD2Clock = (SystemCoreClock >> ((D1CorePrescTable[(RCC->D1CFGR & RCC_D1CFGR_HPRE)>> RCC_D
350
351 }
```

```
352
353
354 /**
355   * @}
356   */
357
358 /**
359   * @}
360   */
361
362 /**
363   * @}
364   */
365 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

A.19 tcp_server.c

```
/**
 * Copyright (c) 2001-2004 Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
 * SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 * OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 *
 * This file is part of and a contribution to the lwIP TCP/IP stack.
 *
 * Credits go to Adam Dunkels (and the current maintainers) of this software.
 *
 * Christiaan Simons rewrote this file to get a more stable echo application.
 *
 **/

 /* This file was modified by ST */

#include <main.h>
#include <tcp_server.h>
#include <SPI.h>
#include "lwip/debug.h"
#include "lwip/stats.h"
#include "lwip/tcp.h"
#include "string.h"

#if LWIP_TCP

static struct tcp_pcb *tcp_server_pcb;

/* ECHO protocol states */
enum tcp_server_states
{
  ES_NONE = 0,
  ES_ACCEPTED,
  ES_RECEIVED,
  ES_CLOSING
};
```

```
57
58  /* structure for maintaing connection infos to be passed as argument
59     to LwIP callbacks*/
60  struct tcp_server_struct
61  {
62    u8_t state;               // current connection state
63    u8_t retries;
64    struct tcp_pcb *pcb;      // pointer on the current tcp_pcb
65    struct pbuf *pin;         // pointer on the received pbuf
66    uint8_t * outdata;        // data to be transmitted
67    int outptr;               // pointer to the data to be transmitted
68    int outlen;               // total size of the data to be transmitted
69  };
70
71  static err_t tcp_server_accept(void *arg, struct tcp_pcb *newpcb, err_t err);
72  static err_t tcp_server_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err);
73  static void tcp_server_error(void *arg, err_t err);
74  static err_t tcp_server_poll(void *arg, struct tcp_pcb *tpcb);
75  static err_t tcp_server_sent(void *arg, struct tcp_pcb *tpcb, u16_t len);
76  static void tcp_server_send(struct tcp_pcb *tpcb, struct tcp_server_struct *es);
77  static void tcp_server_connection_close(struct tcp_pcb *tpcb, struct tcp_server_struct *es);
78
79  /**
80    * @brief  Initializes the tcp echo server
81    * @param  None
82    * @retval None
83    */
84  void tcp_server_init(void)
85  {
86      tcp_server_pcb = tcp_new();                                    // create new tcp pcb (protocol control
87      if (tcp_server_pcb != NULL)
88      {
89          err_t err;
90
91          err = tcp_bind(tcp_server_pcb, IP_ADDR_ANY, TCP_PORT);  // bind pcb to port 2222
92          if (err == ERR_OK)
93          {
94              tcp_server_pcb = tcp_listen(tcp_server_pcb);        // start tcp listening
95              tcp_accept(tcp_server_pcb, tcp_server_accept);      // initialize LwIP tcp_accept callback
96          } else
97          {
98              memp_free(MEMP_TCP_PCB, tcp_server_pcb);            // deallocate the pcb
99          }
100     }
101 }
102
103 void ADS1282_RegsRead(struct tcp_server_struct * es, int adnum)
104 {
105     uint8_t Registers[12];
106
107     es->outdata = (uint8_t *)mem_malloc(13);
108
109     switch(adnum)
110     {
111         case 1: SPI_RegsRead(&hspi1, Registers); break;
112         case 2: SPI_RegsRead(&hspi2, Registers); break;
113         case 3: SPI_RegsRead(&hspi3, Registers); break;
114         case 4: SPI_RegsRead(&hspi4, Registers); break;
115         case 5: SPI_RegsRead(&hspi5, Registers); break;
```

```
116             case 6: SPI_RegsRead(&hspi6, Registers); break;
117             default: SPI_RegsRead(&hspi1, Registers); break;
118         }
119
120         es->outdata[0] = 'R';
121         for( int i = 0; i < 11; i++)
122             es->outdata[i+1] = Registers[i];
123         es->outdata[12] = '\n';
124
125         es->outptr = 0;
126         es->outlen = 13;
127     }
128
129     void STM_Info(struct tcp_server_struct * es){
130         es->outdata = (uint8_t *)mem_malloc(32);
131         sprintf( (char *)es->outdata, "Imode:%i samples:%i\n", mode, samplestosend);
132         es->outptr = 0;
133         es->outlen = strlen((char *)es->outdata);
134     }
135
136     void ADReset()
137     {
138         HAL_GPIO_WritePin(ADRESET_GPIO_Port, ADRESET_Pin, GPIO_PIN_RESET);
139         delay_us(5);
140         HAL_GPIO_WritePin(ADRESET_GPIO_Port, ADRESET_Pin, GPIO_PIN_SET);
141         delay_us(300);          // min 63/fdata + 468/fclk
142     }
143
144     // buffer output format
145     // byte 0:      'D' for data or 'N' for no data (no other bytes)
146     // byte 1..4:   buffer counter
147     // byte 5..6148: data
148     // byte 6149:   '/n'
149     static err_t tcp_server_elab(struct tcp_server_struct * es, struct pbuf *p)
150     {
151         int i,j,regnum,regval,hival,loval,adnum;
152         char cmd;
153
154         for( i = 0; i < p->len; i++)
155         {
156             cmd = ((char *)p->payload)[i];
157             if(cmd == 'r')                      // read registers
158             {
159                 adnum = (int) (((char*)p->payload)[++i]);
160                 ADS1282_RegsRead(es, adnum);
161             } else if(cmd == 'b')               // get buffer
162             {
163                 if(bufin != bufout)             // wait for buffers to be full
164                 {
165                     es->outdata = (uint8_t *)mem_malloc(6150);  // 1(D)+4(counter)+6*4*256(data)+1(e
166                     if(es->outdata != NULL)
167                     {
168                         es->outdata[0] = 'D';
169                         ((uint32_t *)(&(es->outdata[1])))[0] = bufc[bufout];
170                         for(j = 0; j < 6; j++)
171                         {
172                             memcpy(&(es->outdata[5+j*4*256]),buf[j][bufout],4*256);
173                         }
174                         es->outdata[5+6*4*256] = '\n';
```

```
175                     es->outptr = 0;
176                     es->outlen = 6+6*4*256;
177                     bufout = (bufout+1) & 3;
178                 } else
179                 {
180                     es->outdata = (uint8_t *)mem_malloc(1);
181                     es->outdata[0] = 'N';
182                     es->outptr = 0;
183                     es->outlen = 1;
184                 }
185             } else
186             {
187                 es->outdata = (uint8_t *)mem_malloc(1);
188                 es->outdata[0] = 'N';
189                 es->outptr = 0;
190                 es->outlen = 1;
191             }
192         } else if(cmd == 'i')               // STM firmware info
193         {
194             STM_Info(es);
195         } else                              // Commands without output... send 'N'
196         {
197             if(cmd == 'c')                  // calibrate offset
198             {
199             } else if(cmd == 'g')           // calibrate gain
200             {
201             } else if(cmd == 'p')           // wait interrupt nPULS for sending <samplestosend> samples
202             {
203                 samplesent = 0;
204                 bufin = 0;
205                 bufout = 0;
206                 bufind = 0;
207                 mode = 3;
208             } else if(cmd == 'n')           // send now <samplestosend> samples
209             {
210                 samplesent = 0;
211                 bufin = 0;
212                 bufout = 0;
213                 bufind = 0;
214                 mode = 4;
215             } else if(cmd == 'f')           // sending samples continuously
216             {
217                 mode = 0;
218             } else if(cmd == 'w')           // write registers
219             {
220                 regnum = (int) (((char*)p->payload)[++i]);
221                 regval = (int) (((char*)p->payload)[++i]);
222                 adnum = (int) (((char*)p->payload)[++i]);
223 
224                 switch(adnum)
225                 {
226                     case 1: SPI_RegWrite(&hspi1, regnum, regval); break;
227                     case 2: SPI_RegWrite(&hspi2, regnum, regval); break;
228                     case 3: SPI_RegWrite(&hspi3, regnum, regval); break;
229                     case 4: SPI_RegWrite(&hspi4, regnum, regval); break;
230                     case 5: SPI_RegWrite(&hspi5, regnum, regval); break;
231                     case 6: SPI_RegWrite(&hspi6, regnum, regval); break;
232                     default: SPI_RegWrite(&hspi1, regnum, regval); break;
233                 }
```

```
234             } else if(cmd == 's')           // set samples to send
235             {
236                 hival = (int) (((char*)p->payload)[++i]);
237                 loval = (int) (((char*)p->payload)[++i]);
238                 samplestosend = loval + 256 * hival;
239             } else if(cmd == 'x')           // AD reset
240             {
241                 ADReset();
242             }
243             es->outdata = (uint8_t *)mem_malloc(1);
244             es->outdata[0] = 'N';
245             es->outptr = 0;
246             es->outlen = 1;
247         }
248     }
249 
250 //  es->pin = p;                    // store reference to incoming pbuf (chain)
251 /*  sprintf(&es->outdata[es->outlen],"-%3d-%3d-",es->outlen,p->len);
252     es->outlen += 9;
253     memcpy(&es->outdata[es->outlen],p->payload,p->len);
254     for( i = 0; i < p->len; i++)
255     {
256         if(((char *)p->payload)[i] == 'r')
257         {
258             es->outdata[es->outlen + i] = 'z';
259         }
260     }
261     es->outlen += p->len;
262 */
263 
264     pbuf_free(p);
265 }
266 
267 
268 /**
269   * @brief  This function is the implementation of tcp_accept LwIP callback
270   * @param  arg: not used
271   * @param  newpcb: pointer on tcp_pcb struct for the newly created tcp connection
272   * @param  err: not used
273   * @retval err_t: error status
274   */
275 static err_t tcp_server_accept(void *arg, struct tcp_pcb *newpcb, err_t err)
276 {
277     err_t ret_err;
278     struct tcp_server_struct *es;
279 
280     LWIP_UNUSED_ARG(arg);
281     LWIP_UNUSED_ARG(err);
282 
283     tcp_setprio(newpcb, TCP_PRIO_MIN);      // set priority for the newly accepted tcp connectio
284     es = (struct tcp_server_struct *)mem_malloc(sizeof(struct tcp_server_struct));  // allocate
285     if (es != NULL)
286     {
287         es->state = ES_ACCEPTED;
288         es->pcb = newpcb;
289         es->retries = 0;
290         es->pin = NULL;
291         es->outptr = 0;
292         es->outlen = 0;
```

```
293
294         tcp_arg(newpcb, es);                    // pass newly allocated es structure as argument to new
295         tcp_recv(newpcb, tcp_server_recv);      // initialize lwip tcp_recv callback function for newpc
296         tcp_err(newpcb, tcp_server_error);      // initialize lwip tcp_err callback function for newpcb
297         tcp_poll(newpcb, tcp_server_poll, 0);   // initialize lwip tcp_poll callback function for newpc
298
299         ret_err = ERR_OK;
300     } else
301     {
302         tcp_server_connection_close(newpcb, es);// close tcp connection
303         ret_err = ERR_MEM;                      // return memory error
304     }
305     return ret_err;
306 }
307
308 /**
309   * @brief  This function is the implementation for tcp_recv LwIP callback
310   * @param  arg: pointer on a argument for the tcp_pcb connection
311   * @param  tpcb: pointer on the tcp_pcb connection
312   * @param  pbuf: pointer on the received pbuf
313   * @param  err: error information regarding the reveived pbuf
314   * @retval err_t: error code
315   */
316 static err_t tcp_server_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err)
317 {
318     struct tcp_server_struct *es;
319     err_t ret_err;
320
321     LWIP_ASSERT("arg != NULL",arg != NULL);
322
323     es = (struct tcp_server_struct *)arg;
324
325     if (p == NULL)                              // if we receive an empty tcp frame from client => clos
326     {
327         es->state = ES_CLOSING;                 // remote host closed connection
328         if(es->outptr >= es->outlen)
329         {
330             tcp_server_connection_close(tpcb, es); // we're done sending, close connection
331         } else                                  // we're not done yet
332         {
333             tcp_sent(tpcb, tcp_server_sent);    // acknowledge received packet
334             tcp_server_send(tpcb, es);          // send remaining data
335         }
336         ret_err = ERR_OK;
337     } else if(err != ERR_OK)                    // else : a non empty frame was received from client bu
338     {
339         es->outptr = 0;
340         es->outlen = 0;
341         pbuf_free(p);                           // we know that p != NULL -> free received pbuf
342         ret_err = err;
343     } else if(es->state == ES_ACCEPTED)
344     {
345         es->state = ES_RECEIVED;                // first data chunk in p->payload
346         tcp_server_elab(es,p);
347         tcp_sent(tpcb, tcp_server_sent);        // initialize LwIP tcp_sent callback function
348         tcp_server_send(tpcb, es);              // send back the data
349         ret_err = ERR_OK;
350     } else if (es->state == ES_RECEIVED)        // more data received from client
351     {
```

```
352         tcp_server_elab(es,p);
353         tcp_sent(tpcb, tcp_server_sent);         // initialize LwIP tcp_sent callback function
354         tcp_server_send(tpcb, es);               // send data
355         ret_err = ERR_OK;
356     } else if(es->state == ES_CLOSING)
357     {
358         tcp_recved(tpcb, p->tot_len);            // odd case, remote side closing twice, trash da
359         es->outlen = 0;
360         pbuf_free(p);
361         ret_err = ERR_OK;
362     } else
363     {
364         tcp_recved(tpcb, p->tot_len);            // unknown es->state, trash data
365         es->outlen = 0;
366         pbuf_free(p);
367         ret_err = ERR_OK;
368     }
369     return ret_err;
370 }
371
372 /**
373   * @brief  This function implements the tcp_err callback function (called
374   *         when a fatal tcp_connection error occurs.
375   * @param  arg: pointer on argument parameter
376   * @param  err: not used
377   * @retval None
378   */
379 static void tcp_server_error(void *arg, err_t err)
380 {
381     struct tcp_server_struct *es;
382
383     LWIP_UNUSED_ARG(err);
384
385     es = (struct tcp_server_struct *)arg;
386     if (es != NULL)
387     {
388         mem_free(es);           // free es structure
389     }
390 }
391
392 /**
393   * @brief  This function implements the tcp_poll LwIP callback function
394   * @param  arg: pointer on argument passed to callback
395   * @param  tpcb: pointer on the tcp_pcb for the current tcp connection
396   * @retval err_t: error code
397   */
398 static err_t tcp_server_poll(void *arg, struct tcp_pcb *tpcb)
399 {
400     err_t ret_err;
401     struct tcp_server_struct *es;
402
403     es = (struct tcp_server_struct *)arg;
404     if (es != NULL)
405     {
406         if (es->outptr < es->outlen)
407         {
408             tcp_sent(tpcb, tcp_server_sent);
409             tcp_server_send(tpcb, es);
410         } else
```

```
411         {
412             if(es->state == ES_CLOSING)              // no remaining pbuf (chain)
413             {
414                 tcp_server_connection_close(tpcb, es);  // close tcp connection
415             }
416         }
417         ret_err = ERR_OK;
418     } else
419     {
420         tcp_abort(tpcb);                             // nothing to be done
421         ret_err = ERR_ABRT;
422     }
423     return ret_err;
424 }
425 
426 /**
427   * @brief  This function implements the tcp_sent LwIP callback (called when ACK
428   *         is received from remote host for sent data)
429   * @param  None
430   * @retval None
431   */
432 static err_t tcp_server_sent(void *arg, struct tcp_pcb *tpcb, u16_t len)
433 {
434     err_t wr_err = ERR_OK;
435 
436     struct tcp_server_struct *es;
437     int wrsize;
438     int test;
439 
440     LWIP_UNUSED_ARG(len);
441 
442     es = (struct tcp_server_struct *)arg;
443     es->retries = 0;
444 
445     if(es->outptr < es->outlen)
446     {
447         tcp_sent(tpcb, tcp_server_sent);             // still got data to send
448         wrsize = (es->outlen - es->outptr <= tcp_sndbuf(tpcb)) ? es->outlen - es->outptr : tcp_sndbuf(t
449         wr_err = tcp_write(tpcb, &(es->outdata[es->outptr]), wrsize, TCP_WRITE_FLAG_COPY);  // enqueue
450         if (wr_err == ERR_OK)
451         {
452             tcp_recved(tpcb, wrsize);                        // we can read more data now
453             es->outptr += wrsize;
454             if((es->outptr >= es->outlen) && (es->outdata != NULL))
455             {
456                 mem_free(es->outdata);
457                 es->outdata = NULL;
458             }
459         }
460     } else
461     {
462         if(es->state == ES_CLOSING)                      // if no more data to send and client closed co
463             tcp_server_connection_close(tpcb, es);
464     }
465     return wr_err;
466 }
467 
468 /**
469   * @brief  This function is used to send data for tcp connection
```

```
470   * @param  tpcb: pointer on the tcp_pcb connection
471   * @param  es: pointer on state structure
472   * @retval None
473   */
474 static void tcp_server_send(struct tcp_pcb *tpcb, struct tcp_server_struct *es)
475 {
476     err_t wr_err = ERR_OK;
477     int test;
478     int wrsize;
479 
480     if(es->outptr < es->outlen)
481     {
482         wrsize = (es->outlen <= tcp_sndbuf(tpcb)) ? es->outlen : tcp_sndbuf(tpcb);
483         wr_err = tcp_write(tpcb, es->outdata, wrsize, TCP_WRITE_FLAG_COPY); // enqueue data for
484         if (wr_err == ERR_OK)
485         {
486             tcp_recved(tpcb, wrsize);                       // we can read more data now
487             es->outptr += wrsize;
488             if((es->outptr >= es->outlen) && (es->outdata != NULL))
489             {
490                 mem_free(es->outdata);
491                 es->outdata = NULL;
492             }
493         } else if(wr_err == ERR_MEM)
494         {
495             test = 1;
496             // we are low on memory, try later / harder, defer to poll
497         } else
498         {
499             test = 2;
500             // other problem ??
501         }
502     }
503 }
504 
505 /**
506   * @brief  This functions closes the tcp connection
507   * @param  tcp_pcb: pointer on the tcp connection
508   * @param  es: pointer on echo_state structure
509   * @retval None
510   */
511 static void tcp_server_connection_close(struct tcp_pcb *tpcb, struct tcp_server_struct *es)
512 {
513     // remove all callbacks
514     tcp_arg(tpcb, NULL);
515     tcp_sent(tpcb, NULL);
516     tcp_recv(tpcb, NULL);
517     tcp_err(tpcb, NULL);
518     tcp_poll(tpcb, NULL, 0);
519 
520 
521     // delete es structure
522     if (es != NULL)
523     {
524         if(es->outdata != NULL)
525         {
526             mem_free(es->outdata);
527             es->outdata = NULL;
528         }
```

```
529             mem_free(es);
530         }
531 
532         // close tcp connection
533         tcp_close(tpcb);
534     }
535 
536     #endif /* LWIP_TCP */
```

A.20 tcp_server.h

```
/*
 * Copyright (c) 2001-2004 Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
 * SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 * OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 *
 * This file is part of the lwIP TCP/IP stack.
 *
 */
#ifndef __TCP_SERVER_H__
#define __TCP_SERVER_H__

#define TCP_PORT 2222

void tcp_server_init(void);

#endif /* __TCP_SERVER */
```

A.21 STM32H743ZITX_FLASH.ld

```
/*
******************************************************************************
**
**  File        : LinkerScript.ld
**
**  Author      : Auto-generated by STM32CubeIDE
**
**  Abstract    : Linker script for NUCLEO-H743ZI2 Board embedding STM32H743ZITx Device from STM32H7 se
**                      2048Kbytes FLASH
**                      128Kbytes DTCMRAM
**                      64Kbytes ITCMRAM
**                      512Kbytes RAM_D1
**                      288Kbytes RAM_D2
**                      64Kbytes RAM_D3
**
**                Set heap size, stack size and stack location according
**                to application requirements.
**
**                Set memory bank area and size if external memory is used.
**
**  Target      : STMicroelectronics STM32
**
**  Distribution: The file is distributed as is without any warranty
**                of any kind.
**
*****************************************************************************
** @attention
**
** <h2><center>&copy; COPYRIGHT(c) 2019 STMicroelectronics</center></h2>
**
** Redistribution and use in source and binary forms, with or without modification,
** are permitted provided that the following conditions are met:
**   1. Redistributions of source code must retain the above copyright notice,
**      this list of conditions and the following disclaimer.
**   2. Redistributions in binary form must reproduce the above copyright notice,
**      this list of conditions and the following disclaimer in the documentation
**      and/or other materials provided with the distribution.
**   3. Neither the name of STMicroelectronics nor the names of its contributors
**      may be used to endorse or promote products derived from this software
**      without specific prior written permission.
**
** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
** AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
** IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
** DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
** FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
** DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
** SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
** CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
** OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
** OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**
*****************************************************************************
*/

/* Entry Point */
```

```
57  ENTRY(Reset_Handler)
58
59  /* Highest address of the user mode stack */
60  _estack = 0x24080000;    /* end of "RAM_D1" Ram type memory */
61
62  _Min_Heap_Size = 0x200 ;     /* required amount of heap  */
63  _Min_Stack_Size = 0x400 ;    /* required amount of stack */
64
65  /* Memories definition */
66  MEMORY
67  {
68      DTCMRAM (xrw)    : ORIGIN = 0x20000000,  LENGTH = 128K
69      ITCMRAM (xrw)    : ORIGIN = 0x00000000,  LENGTH = 64K
70      RAM_D1  (xrw)    : ORIGIN = 0x24000000,  LENGTH = 512K
71      RAM_D2  (xrw)    : ORIGIN = 0x30000000,  LENGTH = 288K
72      RAM_D3  (xrw)    : ORIGIN = 0x38000000,  LENGTH = 64K
73      FLASH   (rx)     : ORIGIN = 0x8000000,   LENGTH = 2048K
74  }
75
76  /* Sections */
77  SECTIONS
78  {
79    /* The startup code into "FLASH" Rom type memory */
80    .isr_vector :
81    {
82      . = ALIGN(4);
83      KEEP(*(.isr_vector)) /* Startup code */
84      . = ALIGN(4);
85    } >FLASH
86
87    /* The program code and other data into "FLASH" Rom type memory */
88    .text :
89    {
90      . = ALIGN(4);
91      *(.text)           /* .text sections (code) */
92      *(.text*)          /* .text* sections (code) */
93      *(.glue_7)         /* glue arm to thumb code */
94      *(.glue_7t)        /* glue thumb to arm code */
95      *(.eh_frame)
96
97      KEEP (*(.init))
98      KEEP (*(.fini))
99
100     . = ALIGN(4);
101     _etext = .;        /* define a global symbols at end of code */
102   } >FLASH
103
104   /* Constant data into "FLASH" Rom type memory */
105   .rodata :
106   {
107     . = ALIGN(4);
108     *(.rodata)         /* .rodata sections (constants, strings, etc.) */
109     *(.rodata*)        /* .rodata* sections (constants, strings, etc.) */
110     . = ALIGN(4);
111   } >FLASH
112
113   .ARM.extab   : {
114     . = ALIGN(4);
115     *(.ARM.extab* .gnu.linkonce.armextab.*)
```

```
116       . = ALIGN(4);
117     } >FLASH
118
119     .ARM : {
120       . = ALIGN(4);
121       __exidx_start = .;
122       *(.ARM.exidx*)
123       __exidx_end = .;
124       . = ALIGN(4);
125     } >FLASH
126
127     .preinit_array      :
128     {
129       . = ALIGN(4);
130       PROVIDE_HIDDEN (__preinit_array_start = .);
131       KEEP (*(.preinit_array*))
132       PROVIDE_HIDDEN (__preinit_array_end = .);
133       . = ALIGN(4);
134     } >FLASH
135
136     .init_array :
137     {
138       . = ALIGN(4);
139       PROVIDE_HIDDEN (__init_array_start = .);
140       KEEP (*(SORT(.init_array.*)))
141       KEEP (*(.init_array*))
142       PROVIDE_HIDDEN (__init_array_end = .);
143       . = ALIGN(4);
144     } >FLASH
145
146     .fini_array :
147     {
148       . = ALIGN(4);
149       PROVIDE_HIDDEN (__fini_array_start = .);
150       KEEP (*(SORT(.fini_array.*)))
151       KEEP (*(.fini_array*))
152       PROVIDE_HIDDEN (__fini_array_end = .);
153       . = ALIGN(4);
154     } >FLASH
155
156     /* Used by the startup to initialize data */
157     _sidata = LOADADDR(.data);
158
159     /* Initialized data sections into "RAM_D1" Ram type memory */
160     .data :
161     {
162       . = ALIGN(4);
163       _sdata = .;         /* create a global symbol at data start */
164       *(.data)            /* .data sections */
165       *(.data*)           /* .data* sections */
166
167       . = ALIGN(4);
168       _edata = .;         /* define a global symbol at data end */
169
170     } >RAM_D1 AT> FLASH
171
172     /* Uninitialized data section into "RAM_D1" Ram type memory */
173     . = ALIGN(4);
174     .bss :
```

```
175   {
176     /* This is used by the startup in order to initialize the .bss secion */
177     _sbss = .;          /* define a global symbol at bss start */
178     __bss_start__ = _sbss;
179     *(.bss)
180     *(.bss*)
181     *(COMMON)
182 
183     . = ALIGN(4);
184     _ebss = .;          /* define a global symbol at bss end */
185     __bss_end__ = _ebss;
186   } >RAM_D1
187 
188   /* User_heap_stack section, used to check that there is enough "RAM_D1" Ram  type memory left
189   ._user_heap_stack :
190   {
191     . = ALIGN(8);
192     PROVIDE ( end = . );
193     PROVIDE ( _end = . );
194     . = . + _Min_Heap_Size;
195     . = . + _Min_Stack_Size;
196     . = ALIGN(8);
197   } >RAM_D1
198 
199   .lwip_sec (NOLOAD) :
200   {
201     . = ABSOLUTE(0x30040000);
202     *(.RxDecripSection)
203 
204     . = ABSOLUTE(0x30040060);
205     *(.TxDecripSection)
206 
207     . = ABSOLUTE(0x30040200);
208     *(.RxArraySection)
209   } >RAM_D2 AT> FLASH
210 
211   /* Remove information from the compiler libraries */
212   /DISCARD/ :
213   {
214     libc.a ( * )
215     libm.a ( * )
216     libgcc.a ( * )
217   }
218 
219   .ARM.attributes 0 : { *(.ARM.attributes) }
220 }
```

B. Allegati

life.augmented

UM2407
User manual

STM32H7 Nucleo-144 boards (MB1364)

Introduction

The STM32H7 Nucleo-144 boards based on the MB1364 reference board (NUCLEO-H743ZI, NUCLEO-H753ZI) provide an affordable and flexible way for users to try out new concepts and build prototypes, by choosing from the various combinations of performance and power consumption features provided by the STM32H7 Series microcontroller. The ST Zio connector, which extends the Arduino™ Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields. The STM32H7 Nucleo-144 boards do not require any separate probe as they integrate the STLINK-V3 debugger/programmer. The STM32H7 Nucleo-144 boards come with the comprehensive free software libraries and examples available with the STM32Cube MCU Package.

Figure 1. Nucleo-144 board (top view)

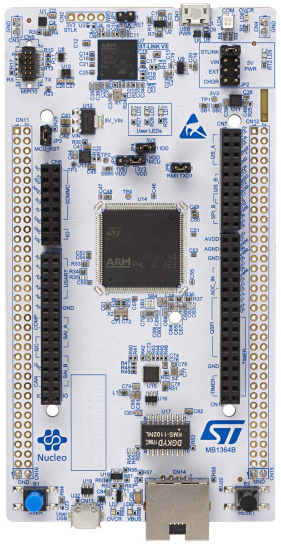
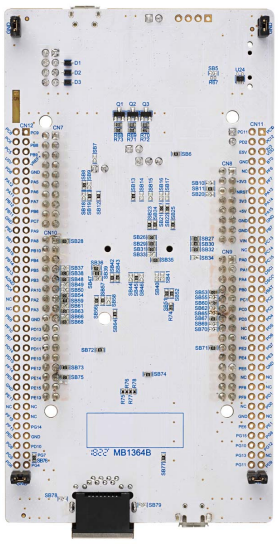


Figure 2. Nucleo-144 board (bottom view)



Pictures are not contractual.

Contents

List of tables

List of figures

1 Features

The STM32H7 Nucleo-144 boards offer the following features:

- STM32H7 Arm®(a) Cortex® core-based microcontroller in LQFP144 package
- Ethernet compliant with IEEE-802.3-2002 (depending on STM32H7 support)
- USB OTG full-speed
- 3 user LEDs
- 2 push-buttons: USER and RESET
- LSE crystal:
  - 32.768 kHz crystal oscillator
- Board connectors:
  - USB with Micro-AB
  - Ethernet RJ45
  - MIPI10
- Board expansion connectors:
  - ST Zio including Arduino™ Uno V3
  - ST morpho
- Flexible power-supply options: ST-LINK USB V_{BUS} or external sources
- On-board STLINK-V3 debugger/programmer with SWD connector:
  - USB re-enumeration capability: virtual COM port, mass storage, debug port
  - STLINK-V3 standalone kit capability
- Comprehensive free software libraries and examples available with the STM32Cube package
- Supported by wide choice of integrated development environments (IDEs) including IAR™, Keil® and GCC-based IDEs

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Ordering information

To order the Nucleo-144 board corresponding to the targeted STM32, use the order code given below in *Table 1*:

Table 1. Ordering information

Order code	Board reference	Target STM32H7	Differentiating feature
NUCLEO-H743ZI	MB1364	STM32H743ZIT6U	– Ethernet – USB OTG FS on Micro.AB connector – ST-LINK/V2-1
NUCLEO-H753ZI		STM32H753ZIT6U	– Ethernet – USB OTG FS on Micro.AB connector – STLINK-V3E – Cryptography

2.1 Product marking

Evaluation tools marked as “ES” or “E” are not yet qualified and therefore not ready to be used as reference design or in production. Any consequences deriving from such usage will not be at ST charge. In no event, ST will be liable for any customer usage of these engineering sample tools as reference design or in production.

“E” or “ES” marking examples of location:

- On the targeted STM32H7 that is soldered on the board (for illustration of STM32H7 marking, refer to the STM32H7 datasheet “Package information” paragraph at the *www.st.com* website).
- Next to the evaluation tool ordering part number that is stuck or silk-screen printed on the board.

These boards feature a specific STM32H7 device version which allows the operation of any stack or library. This STM32H7 device shows a "U" marking option at the end of the standard part number and is not available for sales.

2.2 Codification

The meaning of the codification is explained in *Table 2*.

Table 2. Codification explanation

NUCLEO-XXYYZTN	Description	Example: NUCLEO-H743ZI2
XX	MCU series in STM32 Arm Cortex MCUs	STM32H7 Series
YY	MCU Product line in the series	STM32H743
Z	STM32 package pin count	144 pins

Table 2. Codification explanation

NUCLEO-XXYYZTN	Description	Example: NUCLEO-H743ZI2
T	STM32H7 Flash memory size: -I for 2 Mbytes	2 Mbytes
N	Board version: -void = ST-LINK/V2-1 -2 = STLINK-V3E	STLINK-V3E

This order code is mentioned on a sticker placed on top side of the board.

3 Development environment

3.1 Development toolchains

- Keil® MDK-ARM(a)
- IAR™ EWARM(a)
- GCC-based IDEs

3.2 System requirements

- Windows® OS (7, 8 and 10), Linux® or macOS®(b)
- USB Type-A to Micro-B cable

3.3 Demonstration software

The demonstration software, included in the STM32Cube package, is preloaded in the STM32H7 Flash memory for easy demonstration of the device peripherals in standalone mode. The latest versions of the demonstration source code and associated documentation can be downloaded from the www.st.com/stm32nucleo web page.

a. On Windows only.

b. macOS is a trademark of Apple Inc., registered in the U.S. and other countries.

4 Conventions

Table 3 provides the conventions used for the ON and OFF settings in the present document.

Table 3. ON/OFF conventions

Convention	Definition
Jumper JPx ON	Jumper fitted
Jumper JPx OFF	Jumper not fitted
Solder bridge SBx ON	SBx connections closed by solder or 0 ohm resistor
Solder bridge SBx OFF	SBx connections left open

In this document, for any information that is common to all sales types, the references are noted “STM32H7 Nucleo-144 board” and “STM32H7 Nucleo-144 boards”.

5 Quick start

The STM32H7 Nucleo-144 board is a low-cost and easy-to-use development kit, used to evaluate and start a development quickly with an STM32H7 Series microcontroller in LQFP144 package.

Before installing and using the product, accept the Evaluation Product License Agreement from the www.st.com/epla web page. For more information on the STM32H7 Nucleo-144 and for demonstration software, visit the www.st.com/stm32nucleo web page.

5.1 Getting started

Follow the sequence below to configure the Nucleo-144 board and launch the demonstration application (for components location refer to *Figure 4*):

1. Check jumper position on the board:

Table 4. Jumper configuration

Jumper	Definition	Position	Comment
JP1	STLK_RST	OFF	-
JP3	T_NRST	ON	-
JP5	VDD_MCU power selection	ON [1-2] **(default)**	VDD_MCU supplied with 3V3_VDD
		ON [2-3] (optional)	VDD_MCU supplied with 1V8_VDD
JP4	IDD measurement	ON	MCU current measurement
JP2	Power source selection	ON [1-2]	5V_USB_STLK (from ST-LINK)

2. For the correct identification of the device interfaces from the host PC and before connecting the board, install the Nucleo USB driver available on the www.st.com/stm32nucleo website.
3. To power the board connect the STM32H7 Nucleo-144 board to a PC with a USB cable ‘Type-A to Micro-B’ through the USB connector CN1 on the ST-LINK. As a result, the green LED LD6 (PWR) and LD4 (COM) light up and the red LED LD3 blinks.
4. Press button B1 (left button).
5. Observe the blinking frequency of the three LEDs LD1 to LD3 changes, by clicking on the button B1.
6. The software demonstration and the several software examples, that allow the user to use the Nucleo features, are available at the www.st.com/stm32nucleo web page.
7. Develop an application, using the available examples.

6 Hardware layout and configuration

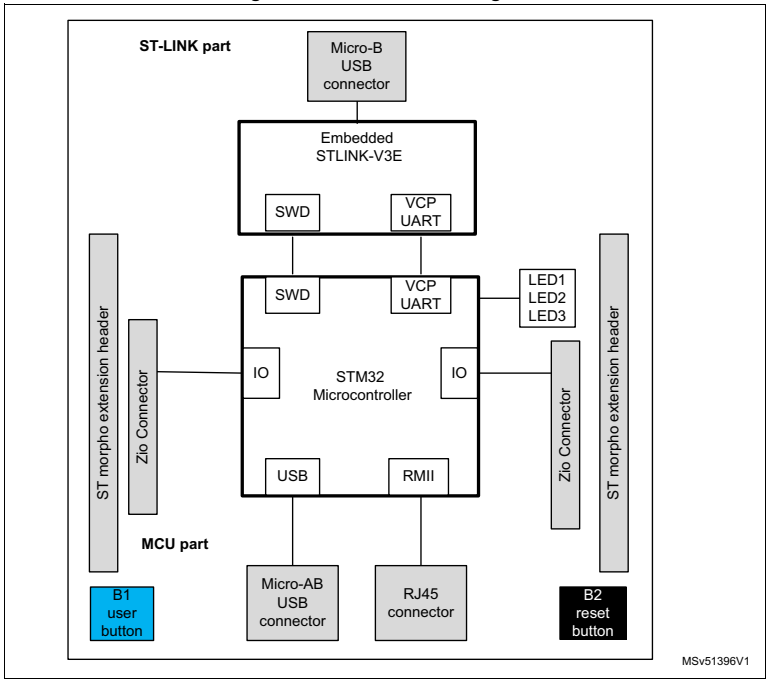
The STM32H7 Nucleo-144 board is designed around the STM32H7 Series microcontrollers in a 144-pin LQFP package.

Figure 3 shows the connections between the STM32H7 and its peripherals (STLINK-V3, push-buttons, LEDs, USB, Ethernet, ST Zio connectors and ST morpho headers).

Figure 4 and *Figure 5* show the location of these features on the STM32H7 Nucleo-144 board.

The mechanical dimensions of the board are showed in *Figure 6* and *Figure 7*.

Figure 3. Hardware block diagram



6.1 Nucleo-144 board layout

Figure 4. Nucleo-144 board top layout

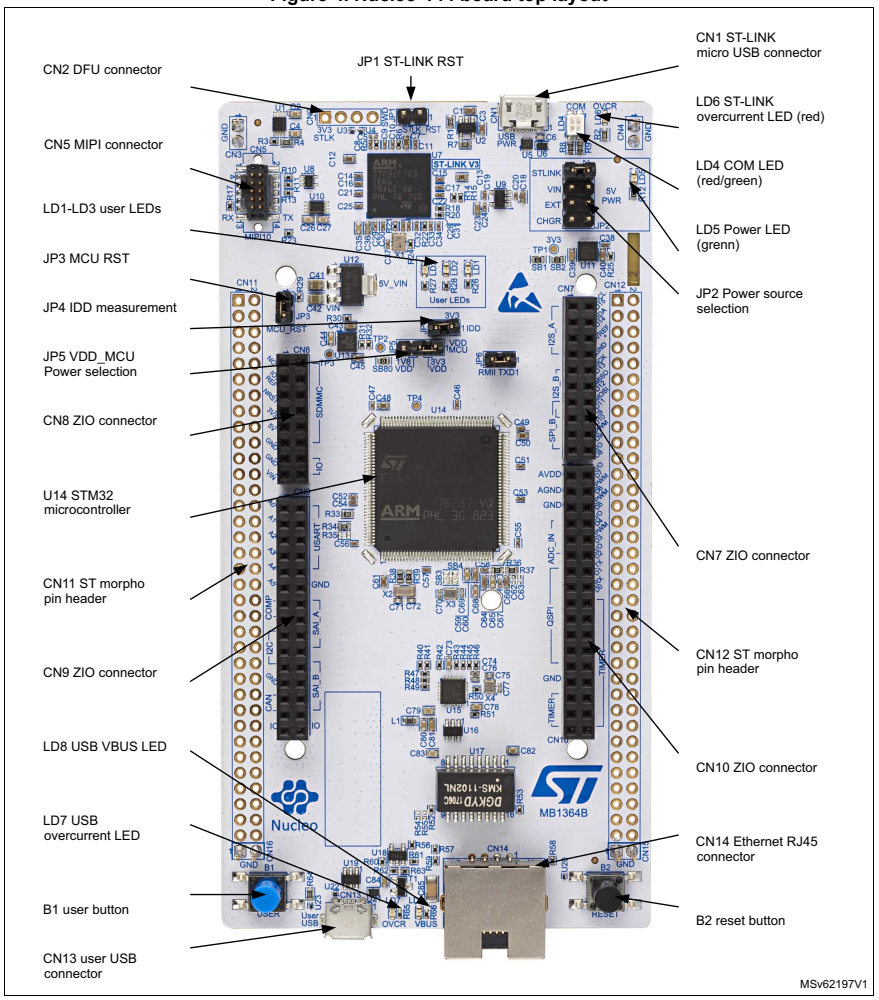
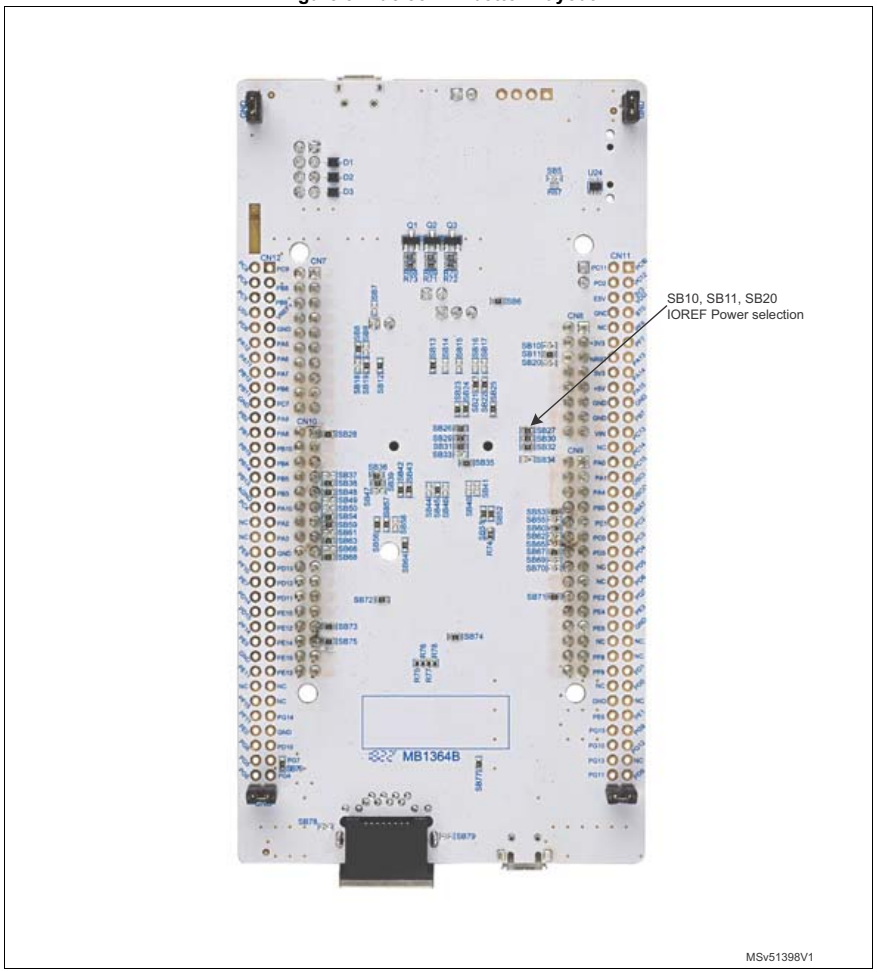


Figure 5. Nucleo-144 bottom layout



6.2 Mechanical drawing

Figure 6. Nucleo-144 board mechanical drawing in millimeter

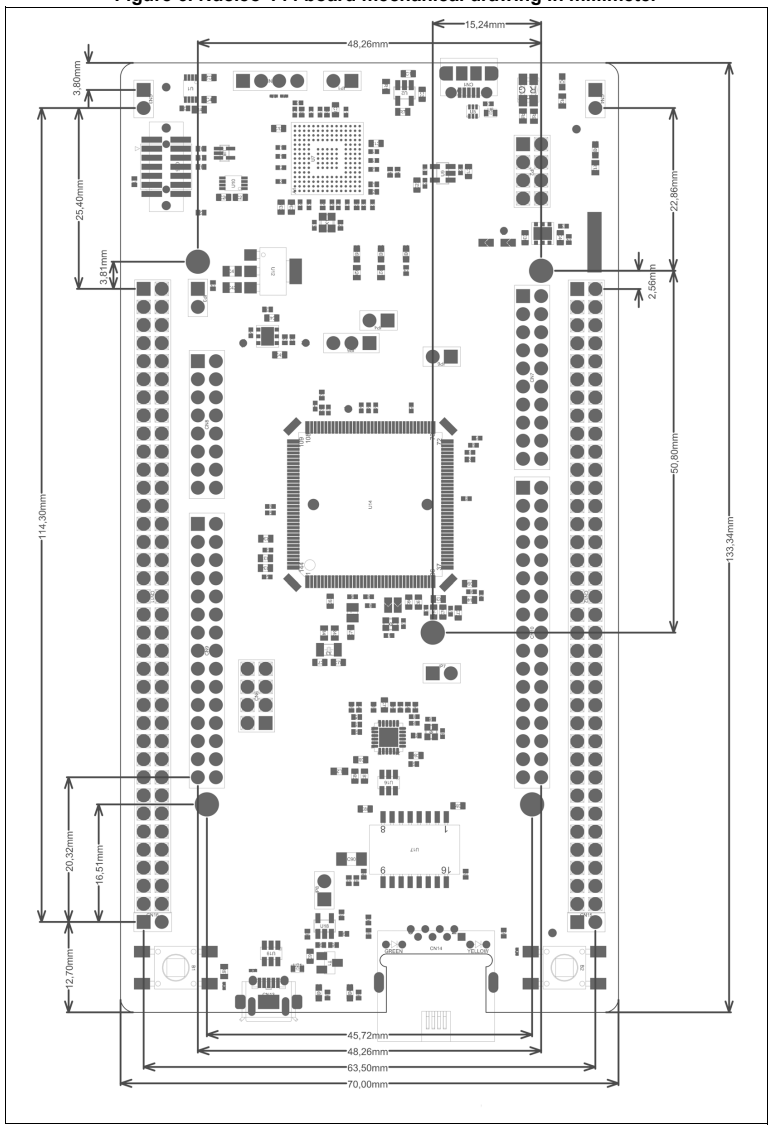
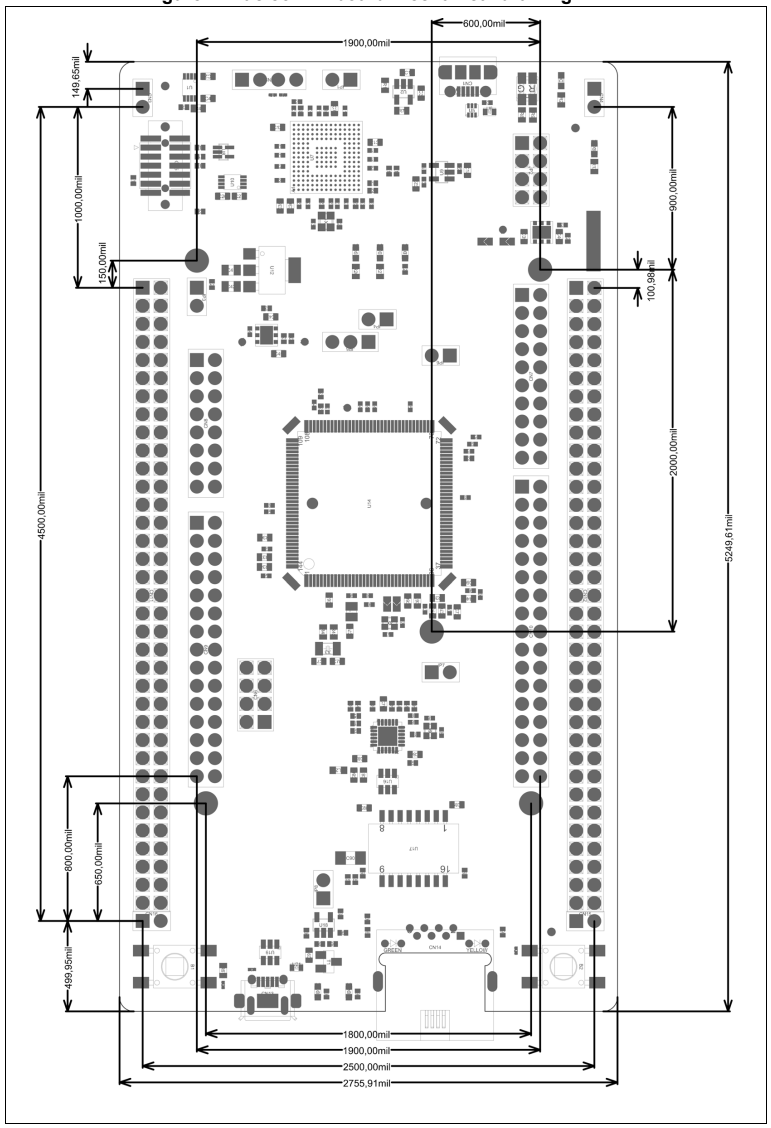


Figure 7. Nucleo-144 board mechanical drawing in mil



6.3 Embedded STLINK-V3

There are two different ways to program or debug the on-board STM32H7 MCU:

- Using the embedded STLINK-V3
- Using an external debug tool connected to CN5 MIPI-10 connector.

The STLINK-V3 programming and debugging tool is integrated in the STM32H7 Nucleo-144 board.

The STLINK-V3 makes the STM32H7 Nucleo-144 board Mbed enabled.

The embedded STLINK-V3 supports only SWD and VCP for STM32H7 devices. For information about debugging and programming features refer to *Overview of ST-LINK derivatives*, Technical note (TN1235), which describes in details all the STLINK/V3 features.

Features supported on STLINK-V3STLINK-V3:

- 5 V power supplied by USB connector (CN1)
- USB 2.0 high-speed-compatible interface
- JTAG/serial wire debugging (SWD) specific features:
  - 3 V to 3.6 V application voltage on the JTAG/SWD interface and 5V tolerant inputs
  - JTAG
  - SWD and serial viewer (SWV) communication
- Direct firmware update feature (DFU) (CN2)
- STDC14 (MIPI10) compatible connector (CN5)
- Status COM LED (LD4) which blinks during communication with the PC
- Fault red LED (LD6) alerting on USB overcurrent request
- 5 V/300 mA output power supply capability (U2) with current limitation and LED
- Green LED ON: 5V enabled (LD5)

6.3.1 Drivers

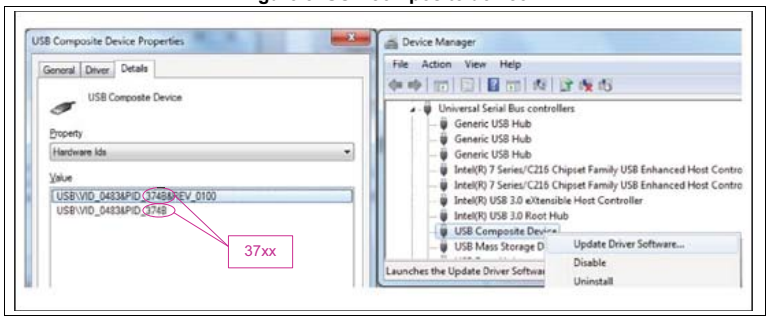
Before connecting the Nucleo-144 board to a Windows 7, Windows 8 or Windows XP PC via USB, a driver for STLINK-V3 must be installed (not require for Windows 10). The driver is automatically installed by the toolset supporting ST-LINK. It is also available from the *www.st.com* website.

In case the STM32H7 Nucleo-144 board is connected to the PC before installing the driver, the PC device manager may report some Nucleo interfaces as “Unknown”.

To recover from this situation, after installing the dedicated driver, the association of “Unknown” USB devices found on the STM32H7 Nucleo-144 board to this dedicated driver, must be updated in the device manager manually.

Note: It is recommended to proceed using USB composite device, as shown in Figure 8.

Figure 8. USB composite device



Note: 37xx = 374E for STLINK-V3 without bridges functions
374F for STLINK-V3 with bridges functions

6.3.2 STLINK-V3 firmware upgrade

The STLINK-V3 embeds a firmware upgrade mechanism for in-situ upgrade through the USB port. As the firmware may evolve during the lifetime of the STLINK-V3 product (for example new functionalities, bug fixes, support for new microcontroller families), it is recommended to keep the STLINK-V3 firmware up to date before starting to use the STM32H7 Nucleo-144 board. The latest version of this firmware is available from the *www.st.com* website.

6.3.3 Using an external debug tool to program and debug the on-boards STM32H7

There are two basic ways to support an external debug tool:

1. Keep the embedded STLINK-V3 running.
   Power on the STLINK-V3 at first until the COM LED lights RED. Then connect your external debug tool through CN5 MIPI-10 debug connector.
2. Set the embedded STLINK-V3 in high impedance state:
   When you state the jumper JP1 (STLK_RST) ON, the embedded STLINK-V3 is in RESET state and all GPIOs are in high impedance, then you can connect your external Debug tool on the debug connector CN5.

Figure 9. Connecting an external debug tool to program the on-board STM32H7

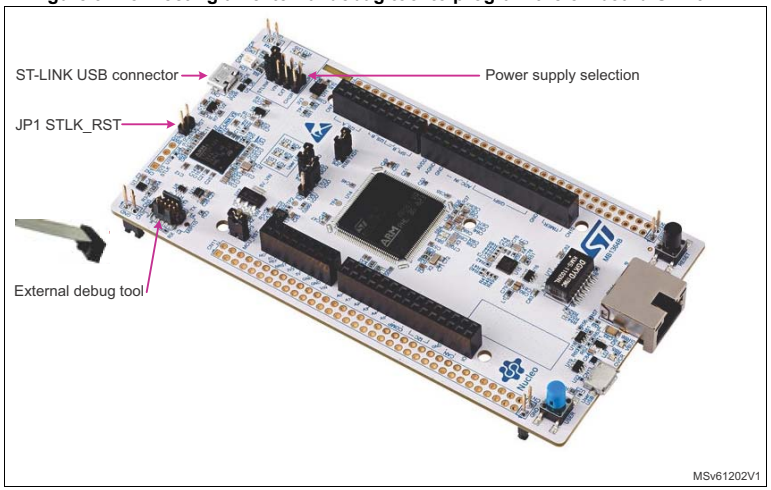


Table 5. MIPI-10 debug connector (CN5)

MIPI-10 Pin	STDC14 Pin	CN5	Designation
-	1	NC	Reserved
-	2	NC	Reserved
1	3	T_VCC	Target VCC
2	4	T_SWDIO	Target SWDIO using SWD protocol or Target JTMS (T_JTMS) using JTAG protocol
3	5	GND	Ground
4	6	T_SWCLK	Target SWCLK using SWD protocol or Target JCLK (T_JCLK) using JTAG protocol
5	7	GND	Ground
6	8	T_SWO	Target SWO using SWD protocol or Target JTDO (T_JTMS) using JTAG protocol
7	9	T_JRCLK	Not used by SWD protocol, Target JRCLK (T_JRCLK) using JTAG protocol, only for specific use
8	10	T_JTDI	Not used by SWD protocol, Target JTDI (T_JTDI) using JTAG protocol, only for external tools
9	11	GNDDetect	GND detect for plug indicator, used on SWD and JTAG neither
10	12	T_NRST	Target NRST using SWD protocol or Target JTMS (T_JTMS) using JTAG protocol

Table 5. MIPI-10 debug connector (CN5) (continued)

MIPI-10 Pin	STDC14 Pin	CN5	Designation
-	13	T_VCP_RX	Target RX used for VCP (must be UART dedicated to bootloader)
-	14	T_VCP_TX	Target TX used for VCP (must be UART dedicated to bootloader)

6.4 Power supply

The power supply can be provided by five different sources:

- A host PC connected to CN1 through a USB cable (default setting)
- An external 7 V to 12 V power supply connected to CN8 pin 15 or CN11 pin 24
- An external 5 V power supply connected to CN11 pin 6
- An external 5 V USB charger (5V_USB_CHGR) connected to CN1
- An external 3.3 V power supply (3V3) connected to CN8 pin 7 or CN11 pin 16

The power supply is provided either by the host PC through the USB cable or by an external source: V_{IN} (7 V to 12 V), E5V (5 V) or +3.3 V power supply pins on CN8 or CN11. In case V_{IN}, E5V or +3.3 V is used to power the Nucleo-144 board, this power source must comply with the standard EN-60950-1: 2006+A11/2009 and must be Safety Extra Low Voltage (SELV) with limited power capability.

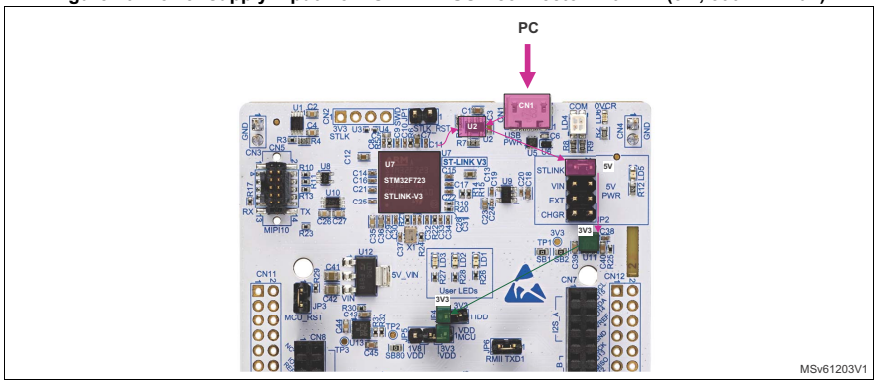
In case the power supply is +3.3 V, the ST-LINK is not powered and cannot be used.

6.4.1 Power supply input from STLINK-V3 USB connector (default setting)

The STM32H7 Nucleo-144 board and shield can be powered from the ST-LINK USB connector CN1 (5 V), by placing a jumper between the pins 1-2 of JP2 "STLINK" (see *Figure 10*).

This is the default setting.

Figure 10. Power supply input from ST-LINK USB connector with PC (5 V, 500 mA max)



If the USB enumeration succeeds, the 5V_ST_LINK power is enabled, by asserting the PWR_ENn signal from STM32F723IEK6 "STLINK V3" (U7). This pin is connected to a power switch STMPS2151STR (U2), which powers the board. The power switch STMPS2151STR (U2) features also a current limitation to protect the PC in case of short-circuit on board. If an overcurrent (more than 500 mA) happens on board, the red LED LD6 is lit.

Nucleo board and its shield on it can be powered from ST-LINK USB connector CN1, but only ST-LINK circuit gets power before USB enumeration, because the host PC only provides 100 mA to the board at that time.

During the USB enumeration, Nucleo board asks for the 500 mA power to the host PC.

- If the host is able to provide the required power, the enumeration finishes by a "SetConfiguration" command and then, the power switch STMPS2151STR is switched ON, the Green LED LD5 is turned ON, thus Nucleo board and its shield on it can consume 500 mA current, but no more.
- If the host is not able to provide the requested current, the enumeration fails. Therefore, the STMPS2151STR power switch (U2) remains OFF and the MCU part including the extension board is not powered. As a consequence, the GREEN LED LD5 remains turned OFF. In this case it is mandatory to use an external power supply.

Warning: In case the maximum current consumption of the STM32H7 Nucleo-144 board and its shield boards exceed 300 mA, it is mandatory to power the STM32H7 Nucleo-144 board, using an external power supply connected to E5V, V_{IN} or +3.3 V.

6.4.2 External power supply input from VIN (7 V to 12 V, 800 mA max)

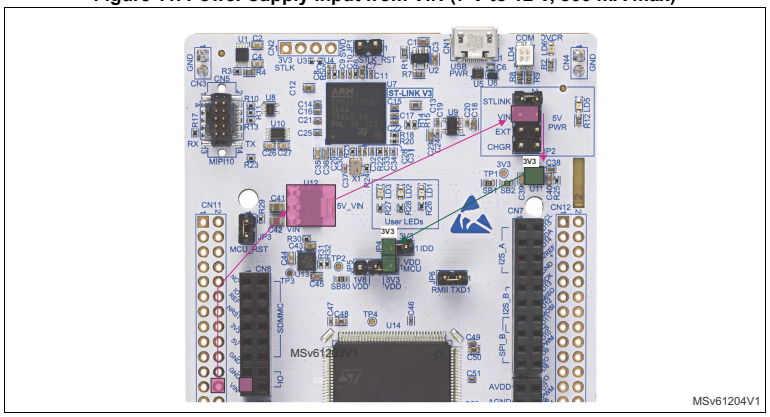
When STM32H7 Nucleo-144 board is power supplied by VIN (see *Table 6* and *Figure 11*), the jumper configuration must be the following: Jumper JP2 on pin 3-4 "VIN"

The Nucleo-144 board and its shield boards can be powered in three different ways from an external power supply, depending on the voltage used. The three power sources are summarized in the *Table 6*.

Table 6. External power sources: VIN (7 V to 12 V)

Input power name	Connector pins	Voltage range	Max current	Limitation
V_{IN}	CN8 pin 15 CN11 pin 24	7 V to 12 V	800 mA	From 7 V to 12 V only and input current capability is linked to input voltage: – 800 mA input current when V_{IN}=7 V – 450 mA input current when 7 V<V_{IN}<9 V – 250 mA input current when 9 V<V_{IN}<12 V

Figure 11. Power supply input from VIN (7 V to 12 V, 800 mA max)



Note: See Section 6.4.6 about debugging when using an external power supply.

6.4.3 External power supply input 5V_EXT (5 V, 500 mA max)

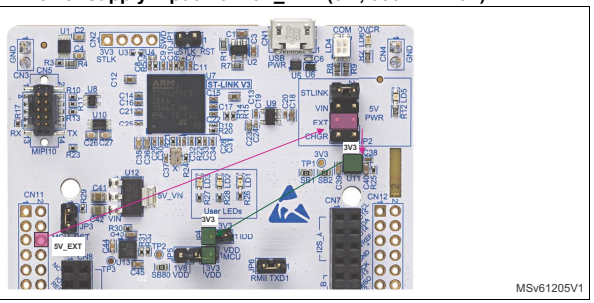
When STM32H7 Nucleo-144 board is power supplied by EXT (see *Table 7* and *Figure 12*), the jumper configuration must be the following: Jumper JP2 on pin 5-6 "EXT"

Table 7. External power sources: 5V_EXT

Input power name	Connector pins	Voltage range	Max current
EXT	CN11 pin 6	4.75 V to 5.25 V	500 mA

Note: See Section 6.4.6 about debugging when using an external power supply.

Figure 12. Power supply input from 5V_EXT (5 V, 500 mA max)



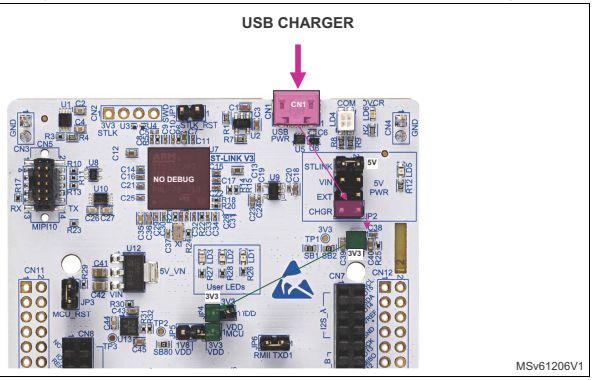
6.4.4 External power supply input from USB CHARGER (5 V)

When STM32H7 Nucleo-144 board is power supplied by a USB charger on CN1 (see *Table 8* and *Table 13*), the jumper configuration must be the following: Jumper JP2 on pin 7-8 "CHGR".

Table 8. External power sources: CHGR (5 V)

Input power name	Connector pins	Voltage range	Max current
CHGR	CN1	5 V	-

Figure 13. Power supply input from ST-LINK USB connector with USB charger (5 V)



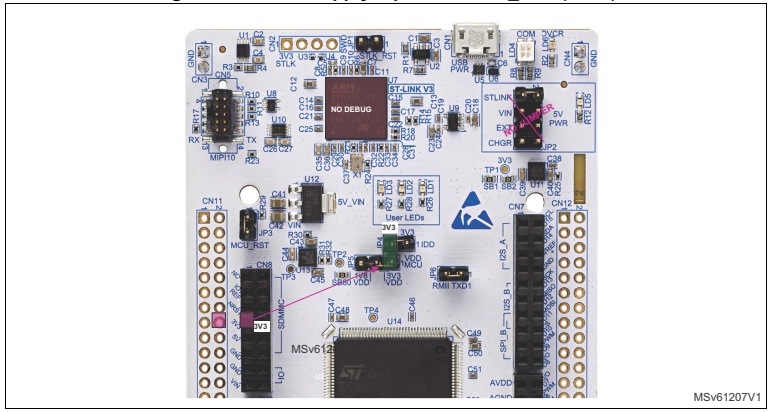
6.4.5 External power supply input from 3V3_EXT (3.3 V)

When the 3.3 V is provided by a shield board, it is interesting to use the 3V3 (CN8 pin 7 or CN11 pin 16) directly as power input (see *Table 9* and *Figure 14*). In this case the programming and debugging features are not available, since the ST-LINK is not powered.

Table 9. External power sources: 3V3_EXT (3.3 V)

Input power name	Connector pins	Voltage range	Max current
3V3	CN8 pin 7 CN11 pin 16	3 V to 3.6 V	1.3 A

Figure 14. Power supply input from 3V3_EXT (3.3 V)



6.4.6 Debugging while using VIN or EXT as an external power supply

When powered by VIN or EXT, it is still possible to use the ST-LINK for programming or debugging only, but it is mandatory to power the board first using VIN or EXT, then to connect the USB cable to the PC. In this way the enumeration succeeds, thanks to the external power source.

The following power-sequence procedure must be respected:

1. Connect jumper JP2 between pin 5 and pin 6 for EXT or between pin 3 and pin 4 for VIN
2. Connect the external power source to VIN or EXT
3. Power on the external power supply 7 V< VIN < 12 V to VIN, or 5 V for EXT
4. Check that the green LED LD5 is turned ON
5. Connect the PC to the USB connector CN1

If this order is not respected, the board may be powered by USB (U5V) first, then by VIN or EXT as the following risks may be encountered:

1. If more than 300 mA current is needed by the board, the PC may be damaged or the current supplied can be limited by the PC. As a consequence, the board is not powered correctly.
2. 300 mA is requested at enumeration so there is risk that the request is rejected and the enumeration does not succeed if the PC cannot provide such current. Consequently, the board is not power supplied (LED LD5 remains OFF).

6.5 Clock sources

6.5.1 HSE clock (high speed external clock)

There are four ways to configure the pins corresponding to the external high-speed clock (HSE):

- MCO from ST-LINK (default): MCO output of ST-LINK is used as input clock. This frequency cannot be changed, it is fixed at 8 MHz and connected to the PF0/PH0-OSC_IN of STM32H7 Series microcontroller. The configuration must be:
  - SB44 and SB46 OFF
  - SB45 ON
  - SB3 and SB4 OFF
- HSE on-board oscillator from X3 crystal (not provided): for typical frequencies and its capacitors and resistors, refer to the STM32H7 Series microcontroller data sheet and to the *Oscillator design guide for STM8AF/AL/S and STM32 microcontrollers* Application note (AN2867) for the oscillator design guide. The X3 crystal has the following characteristics: 25 MHz, 6 pF, 20 ppm. It is recommended to use NX2016SA-25MHz-EXS00A-CS11321 manufactured by NDK. The configuration must be:
  - SB44 and SB46 OFF
  - SB3 and SB4 ON
  - C69 and C70 soldered with 5.6 pF capacitors
  - SB45 OFF

  Oscillator from external PF0/PH0: from an external oscillator through the pin 29 of the CN11 connector. The configuration must be:
  - SB46 ON
  - SB45 OFF
  - SB3 and SB4 OFF
- HSE not used: PF0/PH0 and PF1/PH1 are used as GPIOs instead of as clock. The configuration must be:
  - SB44 and SB46 ON
  - SB45 OFF
  - SB3 and SB4 OFF

6.5.2 LSE clock (low-speed external clock) - 32.768 kHz

There are three ways to configure the pins corresponding to low-speed clock (LSE):

- **On-board oscillator (default):** X2 crystal. Refer to the *Oscillator design guide for STM8AF/AL/S and STM32 microcontrollers* Application note (AN2867) for oscillator design guide for STM32H7 Series microcontrollers. It is recommended to use NX3215SA-32.768kHZ-EXS00A-MU00525 (32.768 kHz, 6 pf load capacitance, 20 ppm) from NDK. The configuration must be:
  - SB40 and SB41 OFF
  - R38 and R39 ON
- **Oscillator from external PC14:** from external oscillator through the pin 25 of CN11 connector. The configuration must be:
  - SB40 and SB41 ON
  - R38 and R39 OFF
- **LSE not used:** PC14 and PC15 are used as GPIOs instead of low-speed clock. The configuration must be:
  - SB40 and SB41 ON
  - R38 and R39 OFF

6.6 Board functions

6.6.1 LEDs

User LD1: a green user LED is connected to the STM32H7 I/O PB0 (SB39 ON and SB47 OFF) or PA5 (SB47 ON and SB39 OFF) corresponding to the ST Zio D13.

User LD2: a yellow user LED is connected to PE1.

User LD3: a red user LED is connected to PB14.

These user LEDs are on when the I/O is HIGH value, and are off when the I/O is LOW.

LD4 COM: the tricolor LED LD4 (green, orange, red) provides information about ST-LINK communication status. LD4 default color is red. LD4 turns to green to indicate that communication is in progress between the PC and the STLINK-V3, with the following setup:

- Slow blinking red/off: at power-on before USB initialization
- Fast blinking red/off: after the first correct communication between PC and STLINK-V3 (enumeration)
- Red LED on: when the initialization between the PC and STLINK-V3 is complete
- Green LED on: after a successful target communication initialization
- Blinking red/green: during communication with target
- Green on: communication finished and successful
- Orange on: communication failure

LD5 PWR: the green LED indicates that the STM32H7 part is powered and +5 V power is available on CN8 pin 9 and CN11 pin 18.

LD6 USB power fault: LD5 indicates that the board power consumption on USB exceeds 500 mA, consequently the user must power the board using an external power supply.

LD7 and LD8 USB FS: refer to *Section 6.6.6: USB OTG FS*.

6.6.2 Push-buttons

B1 USER (blue button): the user button is connected to the I/O PC13 by default (tamper support, SB51 ON and SB58 OFF) or PA0 (Wakeup support, SB58 ON and SB51 OFF) of the STM32H7 Series microcontroller.

B2 RESET (black button): this push-button is connected to NRST and is used to reset the STM32H7 Series microcontroller.

6.6.3 MCU voltage selection: 1V8/3V3

The STM32H7 Nucleo-144 board offers the possibility to supply the STM32H7 Series microcontroller with 1.8 V or 3.3 V. JP5 is used to select the VDD_MCU power level.

- Place the JP5 jumper on 3V3 to supply the MCU with 3V3, connecting pins 1 and 2.
- Place the JP5 jumper on 1V8 to supply the MCU with 1V8, connecting pins 2 and 3.

6.6.4 Current consumption measurement (IDD)

Jumper JP4, labeled IDD, is used to measure the STM32H7 Series microcontroller consumption by removing the jumper and by connecting an ammeter:

- JP4 ON: STM32H7 is powered with 3V3_VDD (default)
- JP4 OFF: an ammeter must be connected to measure the STM32H7 current. If there is no ammeter, the STM32H7 is not powered

Warning: on MB1364 REV.C, "VDD_MCU" is also supplying Ethernet PHY (U15) and debug voltage translation (U1 and U10).

If needed, for low power measurement (for example standby mode), in order to measure only MCU (U7) power consumption, the user must remove the following components: R4, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53, R59, R61, U1, U10, U15 and SB45.

After removing these components, it becomes impossible to use Ethernet, and 1.8 V debug with ST-LINK.

6.6.5 Virtual COM port (VCP): LPUART/USART

The STM32H7 Nucleo-144 board offers the possibility to connect a LPUART or an USART interface to the ST-LINK or to the ST morpho connectors and Arduino™ Uno V3 connectors.

The selection is done by settings the related solder bridges. (Refer to *Table 10* and *Table 11* below).

By default the USART3 communication between the target STM32H7 and the STLINK is enabled, to support the Virtual COM port for the Mbed (SB5 and SB6 ON).

Table 10. USART3 connection

Pin name	Function	Virtual COM port (default configuration)	ST morpho connection
PD8	USART3 TX	SB5 ON and SB7 OFF	SB5 OFF and SB7 ON
PD9	USART3 RX	SB6 ON and SB4 OFF	SB6 OFF and SB4 ON

Table 11. LPUART1 connection

Pin name	Function	Virtual COM port	Arduino D0 and D1	ST morpho connection
PB6	LPUART1 TX	SB9 ON SB8 and SB18 OFF	SB8 ON SB9 and SB18 OFF	SB9 OFF and SB18 OFF
PB7	LPUART1 RX	SB34 ON SB12 and SB68 OFF	SB68 ON SB34 and SB66 OFF	SB12 OFF and SB34 OFF

Hardware connection required for USART bootloader:

The STM32H7x3 embeds USART bootloader. To use the USART bootloader (USART1), hardware modifications are required on the NUCLEO board. Flying wires have to be connected between PD8/PD9 (USART3 available on SB19/SB12) and PB10/PB11 (USART1 available on CN15).

6.6.6 USB OTG FS

The STM32H7 Nucleo-144 board supports USB OTG FS communication via a USB Micro-AB connector (CN13) and USB power switch (U18) connected to V_{BUS}.

Warning: USB Micro–AB connector (CN13) cannot power the Nucleo-144 board. To avoid damaging the STM32H7, it is mandatory to power the Nucleo-144 before connecting a USB cable on CN13. Otherwise there is a risk of current injection on STM32H7 I/Os.

A green LED LD8 lights in one of these cases:

- Power switch (U12) is ON and STM32H7 Nucleo-144 board works as a USB host
- V_{BUS} is powered by another USB host when the STM32H7 Nucleo-144 board works as a USB device.

The red LED LD7 lights if overcurrent occurs when +5 V is enabled on V_{BUS} in USB host mode.

Note: 1.It is recommended to power Nucleo-144 board by an external power supply when using USB OTG or host function.

2.SB76 must be ON when using USB OTG FS.

Table 12. USB pin configuration

Pin name	Function	Configuration when using USB connector	Configuration when using ST morpho connector	Remark
PA8	USB SOF	-	-	Test point TP4
PA9	USB V_{BUS}	SB23 ON	SB23 OFF	-
PA10	USB ID	SB24 ON	SB24 OFF	-
PA11	USB DM	SB21 ON	SB21 OFF	-

Table 12. USB pin configuration (continued)

Pin name	Function	Configuration when using USB connector	Configuration when using ST morpho connector	Remark
PA12	USB DP	SB22 ON	SB22 OFF	-
PD10	USB PWR EN	SB77 ON	SB77 OFF	-
PG7	USB FS OVCR	SB76 ON	SB76 OFF	-

ESD protection part USBLC6-2SC6 is implemented on USB port because all USB pins on STM32H7 are dedicated to USB port protection only on the STM32H7 Nucleo-144 board. USB pin ID is not used.

6.6.7 Ethernet

The STM32H7 Nucleo-144 board supports 10M/100M Ethernet communication by a PHY LAN8742A-CZ-TR (U15) and RJ45 connector (CN14). Ethernet PHY is connected to the STM32H7 Series microcontroller via the RMII interface. 50 MHz clock for the STM32H7 Series microcontroller is generated by the PHY RMII_REF_CLK.

Note: 1.JP6 and SB72 must be ON when using Ethernet.

2.Ethernet PHY LAN8742A should be set in power-down mode (in this mode Ethernet PHY ref clock turns off) to achieve the expected low-power mode current. This is done by configuring Ethernet PHY LAN8742A basic control register (at address 0x00) bit 11 (power down) to '1'. SB57 can also be OFF to get the same effect.

Table 13. Ethernet pin configuration

Pin name	Function	Conflict with ST Zio connector signal	Configuration when using Ethernet	Configuration when using ST Zio or ST morpho connector
PA1	RMII Reference Clock	-	SB57 ON	SB57 OFF
PA2	RMII MDIO	-	SB72 ON	SB72 OFF
PC1	RMII MDC	-	SB64 ON	SB64 OFF
PA7	RMII RX Data Valid	-	SB31 ON	SB31 OFF
PC4	RMII RXD0	-	SB36 ON	SB36 OFF
PC5	RMII RXD1	-	SB29 ON	SB29 OFF
PG11	RMII TX Enable	-	SB27 ON	SB27 OFF
PG13	RXII TXD0	-	SB30 ON	SB30 OFF
PB13	RMII TXD1	I2S_A_CK	JP6 ON	JP6 OFF

6.7 Solder bridges and jumpers

SBxx can be found on top layer and SB1xx can be found on bottom layer of the Nucleo-144 board.

Table 14. Solder bridge and jumper configuration

Bridge	State(1)	Description
SB1 (3V3_PER)	**ON**	Peripheral power 3V3_PER is connected to 3V3.
	OFF	Peripheral power 3V3_PER is not connected.
SB2 (3V3)	ON	Output of voltage regulator ST1L05CPU33R is connected to 3V3.
	OFF	Output of voltage regulator ST1L05CPU33R is not connected.
SB80 (1V8_VDD)	ON	Output of voltage regulator ST1L05BPUR is connected to 1V8_VDD.
	OFF	Output of voltage regulator ST1L05BPUR is not connected.
SB6	ON	Input of voltage regulator ST1L05BPUR is connected to 3V3_VDD.
	OFF	Input of voltage regulator ST1L05BPUR is not connected.
SB12, SB19 (ST-LINK-USART)	**ON**	PG9 and PG14 on ST-LINK STM32F723IEK6 are connected to PD8 and PD9 to enable virtual COM port for Mbed support. Thus PD8 and PD9 on ST morpho connectors cannot be used.
	OFF	PG9 and PG14 on ST-LINK STM32F723IEK6 are disconnected to PD8 and PD9 on STM32H7.
JP1 (ST-LINK_RST)	**OFF**	No incidence on ST-LINK STM32F723IEK6 NRST signal.
	ON	ST-LINK STM32F723IEK6 signal is connected to GND (ST-LINK reset to reduce power consumption).
SB32 (SWO)	**ON**	SWO signal of the STM32H7 (PB3) is connected to ST-LINK SWO input. (SB26 must be removed)
	OFF	SWO signal of STM32H7 is not connected.
JP3 (NRST)	**ON**	Board RESET signal (NRST) is connected to ST-LINK reset control I/O (T_NRST).
	OFF	Board RESET signal (NRST) is not connected to ST-LINK reset control I/O (T_NRST).
SB10, SB11, SB20 (IOREF)	**OFF, ON, OFF**	IOREF is connected to VDD_MCU.
	ON, OFF, OFF	IOREF is connected to 3V3_PER.
	OFF, OFF, ON	IOREF is connected to 3V3.
SB14 (SDMMC_D0), SB15 (SDMMC_D1)	ON	These pins are connected to ST morpho connector CN12.
	OFF	These pins are disconnected from ST morpho connector CN12 to avoid stub of SDMMC data signals on PCB.

Table 14. Solder bridge and jumper configuration (continued)

Bridge	State(1)	Description
SB39, SB47 (LD1-LED)	**ON, OFF**	Green user LED LD1 is connected to PB0.
	OFF,ON	Green user LED LD1 is connected to D13 of Arduino signal (PA5).
	OFF, OFF	Green user LED LD1 is not connected.
	ON, ON	Forbidden
SB33, SB35 (D11)	OFF, ON	D11 (Pin 14 of CN7) is connected to STM32H7 PB5 (SPI_A_MOSI/ TIM_D_PWM2)
	ON,OFF	D11 (Pin 14 of CN7) is connected to STM32H7 PA7 (SPI_A_MOSI/ TIM_E_PWM1)
SB40,SB41 (X2 crystal)	**OFF, OFF**	PC14, PC15 are not connected to ST morpho connector CN11. (X2 used to generate 32 kHz clock).
	ON, ON	PC14, PC15 are connected to ST morpho connector CN11. (R38 and R39 should be removed).
SB44 (PF1/PH1) SB46 (PF0/PH0) (Main clock)	ON, OFF	PF0/PH0 is not connected to ST morpho connector CN11 PF1/PH1 is connected to ST morpho connector CN11 (MCO is used as main clock for STM32H7 on PF0/PH0 – SB45 ON).
	OFF, OFF	PF0/PH0, PF1/PH1 are not connected to ST morpho connector CN11 (X3, C69, C70, SB3 and SB4 provide a clock as shown in *Section Appendix A: Electrical schematics*. In this case SB45 must be removed).
	ON, ON	PF0/PH0 and PF1/PH1 are connected to ST morpho connector CN11. (SB3, SB4 and SB45 must be removed).
SB45 (STLK_MCO)	**ON**	MCO of ST-LINK (STM32F723IEK6) is connected to PF0/PH0 of STM32H7.
	OFF	MCO of ST-LINK (STM32F723IEK6) is not connected to PF0/PH0 of STM32H7.
SB3, SB4 (external 25M crystal)	**OFF, OFF**	PF0/PH0 and PF1/PH1 are not connected to external 25 MHz crystal X3.
	ON, ON	PF0/PH0 and PF1/PH1 are connected to external 25 MHz crystal X3.
SB52 (V_{BAT})	**ON**	V_{BAT} pin of STM32H7 is connected to V_{DD_MCU}.
	OFF	V_{BAT} pin of STM32H7 is not connected to V_{DD_MCU}.
SB51, SB58 (B1-USER)	ON, OFF	B1 push-button is connected to PC13.
	OFF,ON	B1 push-button is connected to PA0 (Set SB51 OFF if ST Zio connector is used).
	OFF,OFF	B1 push-button is not connected.

Table 14. Solder bridge and jumper configuration (continued)

Bridge	State(1)	Description
SB75 (PA0)	**ON**	PA0 is connected to ST Zio connector (Pin 29 of CN10).
	OFF	PA0 is not connected to ST Zio connector (Pin 29 of CN10).
RMII Signals SB57 (PA1), SB64 (PC1), SB72 (PA2), SB36 (PC4), SB29 (PC5), SB30 (PG13), SB27 (PG11), SB31 (PA7), JP6 (PB13)	**ON**	These pins are used as RMII signals and connected to Ethernet PHY. (SB7 must be removed) These pins must not be used on ST morpho or ST Zio connectors.
	OFF	These pins can be used as GPIOs on ST morpho connectors. PB13 can be used as I2S_A_CK on ST Zio (Pin 5 of CN7) if not used on ST morpho.
SB74 (Ethernet nRST) RMII Signal	**ON**	NRST of STM32H7 is connected to Ethernet PHY (U15).
	OFF	NRST of STM32H7 is not connected to Ethernet PHY (U15).
SB76 (PG7)	ON	USB overcurrent alarm is connected.
	OFF	USB overcurrent alarm is not connected. PG7 is used as GPIO on ST morpho connector (CN12).
SB77 (PD10)	ON	PD10 is connected to Enable for Power switch (U18) to control V_{BUS}.
	OFF	PD10 is used as GPIO on ST morpho connector (CN12).
SB23 (PA9)	ON	PA9 is connected to USB V_{BUS}.
	OFF	PA9 is not connected to USB V_{BUS}. PA9 is used as GPIO on ST morpho connector (CN12).
SB24 (PA10)	ON	PA10 is connected to USB ID.
	OFF	PA10 is not connected to USB ID. PA10 is used as GPIO on ST morpho connector (CN12).
SB21 (PA11), SB22 (PA12)	ON	These pins are used as D- and D+ on USB connector CN13. (SB16 and SB17 must be OFF).
	OFF	These pins are used as GPIOs on ST morpho connectors.
SB13	ON	VDD33_USB_1 is connected to 3V3_VDD.
	OFF	VDD33_USB_1 is not supplied.
SB25	ON	VDD_MMC_1 is connected to VDD_MCU.
	OFF	VDD_MMC_1 is not supplied.
SB59 (PG6)	ON	PG6 is connected to QSPI_CS (SB61 must be OFF).
	OFF	PG6 is used as GPIO on ST morpho connector (CN12).
SB63 (PB2)	ON	PB2 is connected to QSPI_CLK. (SB69 must be OFF)
	OFF	PB2 is not connected to QSPI_CLK and can be used as COMP1_INP (SB69 ON) or used as GPIO on ST morpho connector CN12. (SB69 OFF)

Table 14. Solder bridge and jumper configuration (continued)

Bridge	State(1)	Description
SB71, SB73 (PE6)	ON, OFF	PE6 is connected to SAI_A_SD (D59 of CN9)
	OFF, ON	PE6 is connected to TIMER_A_BKIN2 (D38 of CN10)
SB67 (PE2)	ON	PE2 is connected to SAI_A_MCLK (D56 of CN9). QSPI_BK1_IO2 cannot be used (D31 of CN10).
	OFF	PE2 is used as QSPI_BK1_IO2 (D31 of CN10).
SB53 (PC2) and SB60 (PF10)	ON	ADC_IN are connected to A4 and A5 (pin 9 and 11) on ST Zio connector CN9. Thus SB55 and SB62 must be OFF
	OFF	ADC_IN are connected to A4 and A5 (pin 9 and 11) on ST Zio connector CN9. Thus SB55 and SB62 can be ON (I^2C)
SB65 (PF11)	OFF	On NUCLEO-H743ZI2 and NUCLEO-H753ZI, PF11 is used only as GPIO on ST morpho connector (CN12) and must not be used as ADC_IN.
I^2C SB55 (PB9) and SB62 (PB8)	ON	PB9 and PB8 (I2C) are connected to A4 and A5 (pin 9 and 11) on ST Zio connector CN9. Thus SB60 and SB53 must be OFF
	OFF	PB9 and PB8 (I2C) are not connected to A4 and A5 (pin 9 and 11) on ST Zio connector CN9.
SB28 and SB70 (PE9)	ON, OFF	PE9 is used as TIMER_A_PWM1 (Pin 4) on ST Zio connector CN10.
	OFF, ON	PE9 is used as COMP2_INP (Pin 15) on ST Zio connector CN9.
SB37 (PF12) and SB38 (PF4)	OFF, ON	ADC_IN is connected to A6 (pin 7) on ST Zio connector CN10. PF12 must not be used as ADC_IN (SB37 must be OFF)
SB48 (PF5) and SB49 (PF13)	ON, OFF	ADC_IN is connected to A7 (pin 9) on ST Zio connector CN10. PF13 must not be used as ADC_IN (SB49 must be OFF)
SB50 (PF14) and SB54 (PF6)	OFF, ON	ADC_IN is connected to A8 (pin 11) on ST Zio connector CN10. PF14 must not be used as ADC_IN (SB50 must be OFF)
SB5	OFF	NUCLEO-H743ZI2 and NUCLEO-H753ZI support 1V8 and 3V3 for VDD_MCU. Thus U10 level shifter is needed and SB5 must be OFF.
	ON	If the MCU is supplied with 3V3, U10 can be by-passed and SB5 can be ON.

1. Default SBx state is shown in bold.

All the other solder bridges present on the STM32H7 Nucleo-144 board are used to configure several I/Os and power supply pins for compatibility of features and pinout with the target STM32H7 supported.

The STM32H7 Nucleo-144 board is delivered with the solder bridges configured, according to the target STM32H7 supported.

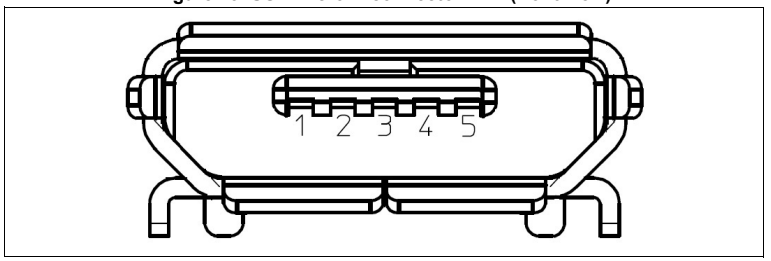
7 Board connectors

Several connectors are implemented on the STM32H7 Nucleo-144 board.

7.1 STLINK-V3 USB Micro-B connector CN1

The USB Micro-B connector CN1 is used to connect embedded STLINK-V3 to the PC for the programming and debugging purposes.

Figure 15. USB Micro-B connector CN1 (front view)



The related pinout for the USB ST-LINK connector is listed in *Table 15*.

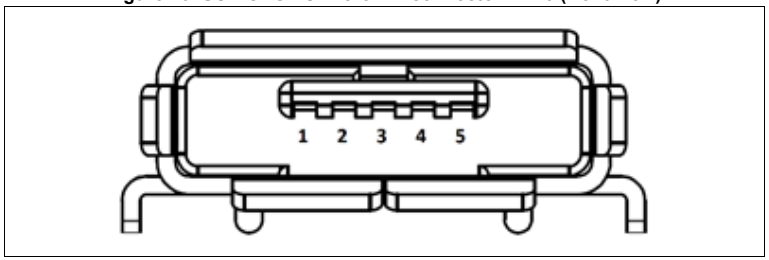
Table 15. USB Micro-B connector pinout

Connector	Pin number	Pin name	Signal name	ST-LINK MCU pin	Function
CN1	1	VBUS	5V_USB_CHGR	-	5 V power
	2	DM	USB_DEV_HS_CN_N	PB14	USB differential pair N
	3	DP	USB_DEV_HS_CN_P	PB15	USB differential pair P
	4	ID	-	-	-
	5	GND	-	-	GND

7.2 USB OTG FS connector CN13

An USB OTG Full Speed communication link is available at USB Micro-AB receptacle connector CN13. Micro-AB receptacle enables USB Host and USB Devices features.

Figure 16. USB OTG FS Micro-AB connector CN13 (front view)



The related pinout for the USB OTG FS connector is listed in *Table 16*.

Table 16. USB OTG FS Micro-AB connector pinout

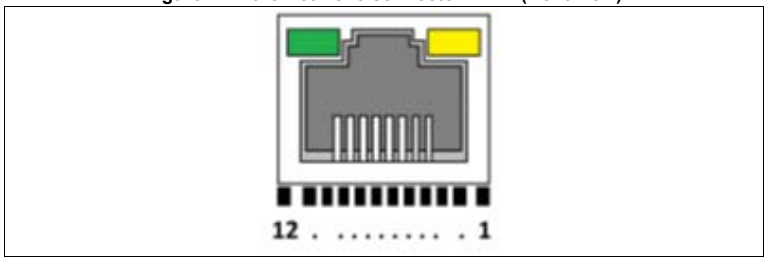
Connector	Pin number	Pin name	Signal name	MCU pin	Function
CN13	1	VBUS	USB_FS_VBUS	PA9	5 V power
	2	DM	USB_FS_N	PA11	USB differential pair M
	3	DP	USB_FS_P	PA12	USB differential pair P
	4	ID	USB_FS_ID	PA10	-
	5	GND	-	-	GND

7.3 Ethernet RJ45 connector CN14

The STM32H7 Nucleo-144 board supports 10Mbps/100Mbps Ethernet communication with the U15 LAN8742A-CZ-TR PHY from MICROCHIP and CN14 integrated RJ45 connector. The Ethernet PHY is connected to the MCU via the RMII interface.

The 25 MHz clock for the PHY is generated by oscillator X4. The 50 MHz clock for the MCU (derived from the 25 MHz crystal oscillator) is provided by the RMII_REF_CLK of the PHY.

Figure 17. Ethernet RJ45 connector CN14 (front view)



1. Green LED: Ethernet traffic
2. Amber LED: Ethernet connection

The related pinout for the Ethernet connector is listed in *Table 17*.

Table 17. Ethernet connector pinout

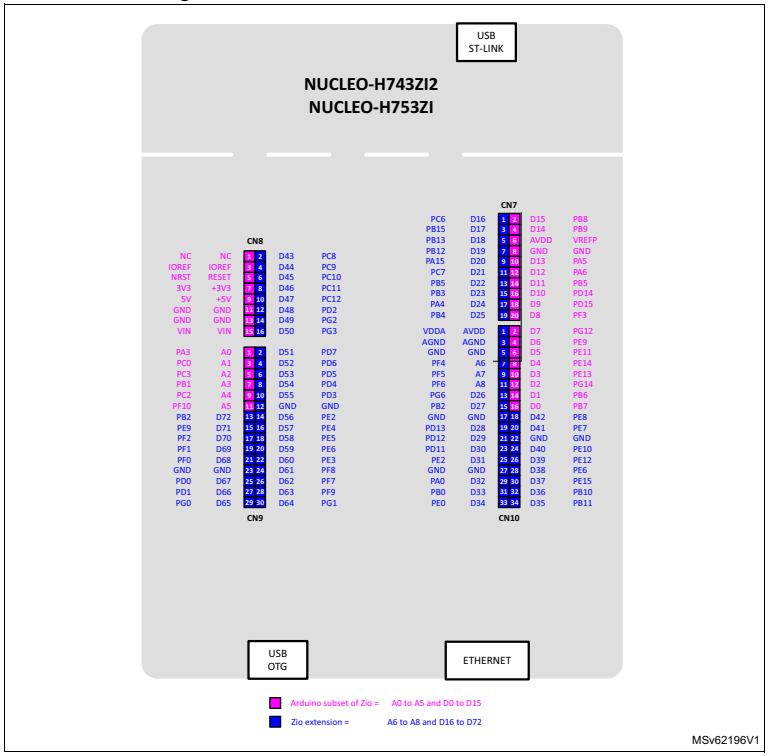
Connector	Pin number	Description	MCU pin	Pin number	Description	MCU pin
CN14	1	TX+	-	7	NC	-
	2	TX-	-	8	NC	-
	3	RX+	-	9	K, yellow LED	-
	4	NC	-	10	A, yellow LED	-
	5	NC	-	11	K, green LED	-
	6	RX-	-	12	A, green LED	-

8 Extension connectors

8.1 ST Zio connectors

For each STM32H7 Nucleo-144 board, the following figures show the signals connected by default to the ST Zio connectors (CN7, CN8, CN9, CN10), including the support for Arduino Uno V3.

Figure 18. NUCLEO-H743ZI2 and NUCLEO-H753ZI



CN7, CN8, CN9 and CN10 are female on top side and male on bottom side connectors. They include support for Arduino Uno V3. Most shields designed for Arduino Uno V3 can fit to the STM32H7 Nucleo-144 board.

To cope with Arduino Uno V3, apply the following modifications:

- SB55 and SB62 should be ON
- SB53/60/65 should be OFF to connect I^2C on A4 (pin 9) and A5 (pin 11 of CN9).

Caution:1 The I/Os of STM32H7 Series microcontroller are 3.3 V compatible instead of 5 V for Arduino Uno V3.

Caution:2 R37 should be removed before implementing Arduino shield with V_{REF+} power being provided on CN7 pin 6. Refer to *Table 14: Solder bridge and jumper configuration* for details on R37.

NUCLEO-H743ZI2, NUCLEO-H753ZI pin assignments

Table 18. CN7 ZIO included Arduino™ connector pinout(1)

Pin	Pin name	Signal name	STM32H7 pin	MCU Function	Pin	Pin name	Signal name	STM32H7 pin	MCU Function
1	**D16**	I2S_A_MCK	PC6	I2S_2	2	**D15**	I2C_A_SCL	PB8	I2C_1_SCL
3	**D17**	I2S_A_SD	PB15	I2S_2	4	**D14**	I2C_A_SDA	PB9	I2C_1_SDA
5	**D18**	I2S_A_CK	PB13(2)	I2S_2	6	**VREFP**	VREFP	-	VDDA/VREFP
7	**D19**	I2S_A_WS	PB12	I2S_2	8	**GND**	GND	-	-
9	**D20**	I2S_B_WS	PA15	I2S_3	10	**D13**	SPI_A_SCK	PA5	SPI1_SCK
11	**D21**	I2S_B_MCK	PC7	I2S_3	12	**D12**	SPI_A_MISO	PA6	SPI1_MISO
13	**D22**	I2S_B_SD/ SPI_B_MOSI	PB5	I2S_3/ SPI3	14	**D11**	SPI_A_MOSI / TIM_E_PWM1	PB5(3)	SPI1_MOSI/ TIM3_CH2
15	**D23**	I2S_B_CK/ SPI_B_SCK	PB3	I2S_3/ SPI3	16	**D10**	SPI_A_CS / TIM_B_PWM3	PD14	SPI1_CS/ TIM4_CH3
17	**D24**	SPI_B_NSS	PA4	SPI3	18	**D9**	TIM_B_PWM2	PD15	TIM4_CH4
19	**D25**	SPI_B_MISO	PB4	SPI3	20	**D8**	I/O	PF3	-

1. For more details refer to *Table 14: Solder bridge and jumper configuration*.
2. PB13 is used as I2S_A_CK and connected to CN7 pin 5. If JP6 is ON, it is also connected to Ethernet PHY as RMII_TXD1. In this case only one function of the Ethernet or I2S_A must be used.
3. PA7 is used as D11 and connected to CN7 pin 14. If SB31 is ON, it is also connected to both Ethernet PHY as RMII_CRS_DV. In this case only one function of the Ethernet or D11 must be used.

Table 19. CN8 ZIO included Arduino™ connector pinout

Pin	Pin name	Signal name	STM32H7 pin	MCU Function	Pin	Pin name	Signal name	STM32H7 pin	MCU Function
1	**NC**	NC	-	-	2	**D43**	SDMMC_D0	PC8	SDMMC
3	**IOREF**	IOREF	-	3.3 V Ref	4	**D44**	SDMMC_D1 I2S_A_CKIN	PC9	SDMMC I2S_CKIN
5	**NRST**	NRST	NRST	RESET	6	**D45**	SDMMC_D2	PC10	SDMMC

Table 19. CN8 ZIO included Arduino™ connector pinout (continued)

Pin	Pin name	Signal name	STM32H7 pin	MCU Function	Pin	Pin name	Signal name	STM32H7 pin	MCU Function
7	**3V3**	3V3	-	3.3 V input/output	8	**D46**	SDMMC_D3	PC11	SDMMC
9	**5V**	5V	-	5 V output	10	**D47**	SDMMC_CK	PC12	SDMMC
11	**GND**	GND	-	ground	12	**D48**	SDMMC_CMD	PD2	SDMMC
13	**GND**	GND	-	ground	14	**D49**	I/O	PG2	-
15	**VIN**	VIN	-	Power input	16	**D50**	I/O	PG3	-

Table 20. CN9 ZIO included Arduino™ connector pinout

Pin	Pin name	Signal name	STM32H7 pin	MCU Function	Pin	Pin name	Signal name	STM32H7 pin	MCU Function
1	**A0**	**ADC**	PA3	ADC12_INP15	2	**D51**	USART_B_SCLK	PD7	USART_2
3	**A1**	**ADC**	PC0	ADC123_INP10	4	**D52**	USART_B_RX	PD6	USART_2
5	**A2**	**ADC**	PC3	ADC12_INP13	6	**D53**	USART_B_TX	PD5	USART_2
7	**A3**	**ADC**	PB1	ADC12_INP5	8	**D54**	USART_B_RTS	PD4	USART_2
9	**A4**	**ADC**	PC2/ PB9	ADC123_INP12/ I2C1_SDA	10	**D55**	USART_B_CTS	PD3	USART_2
11	**A5**	**ADC**	PF10/ PB8	ADC3_INP6/ I2C1_SCL	12	**GND**	GND	-	-
13	**D72**	**COMP1_INP**	PB2	COMP1_INP	14	**D56**	SAI_A_MCLK	PE2(1)	SAI_1_A
15	**D71**	**COMP2_INP**	PE0	COMP2_INP	16	**D57**	SAI_A_FS	PE4	SAI_1_A
17	**D70**	**I2C_B_SMBA**	PF2	I2C2	18	**D58**	SAI_A_SCK	PE5	SAI_1_A
19	**D69**	**I2C_B_SCL**	PF1	I2C2	20	**D59**	SAI_A_SD	PE6	SAI_1_A
21	**D68**	**I2C_B_SDA**	PF0	I2C2	22	**D60**	SAI_B_SD	PE3	SAI_1_B
23	**GND**	**GND**	-	-	24	**D61**	SAI_B_SCK	PF8	SAI_1_B
25	**D67**	**CAN_RX**	PD0	CAN_1	26	**D62**	SAI_B_MCLK	PF7	SAI_1_B

Table 20. CN9 ZIO included Arduino™ connector pinout (continued)

Pin	Pin name	Signal name	STM32H7 pin	MCU Function	Pin	Pin name	Signal name	STM32H7 pin	MCU Function
27	**D66**	**CAN_TX**	PD1	CAN_1	28	**D63**	SAI_B_FS	PF9	SAI_1_B
29	**D65**	**I/O**	PG0	-	30	**D64**	I/O	PG1	-

1. PE2 is connected to both CN9 pin 14 (SAI_A_MCLK) and CN10 pin 25 (QSPI_BK1_IO2). Only one function must be used at one time.

Table 21. CN10 ZIO included Arduino™ connector pinout

Pin	Pin name	Signal name	STM32H7 pin	MCU Function	Pin	Pin name	Signal name	STM32H7 pin	MCU Function
1	**AVDD**	VDDA	-	Analog VDD	2	**D7**	I/O	PG12	I/O
3	**AGND**	AGND	-	Analog GND	4	**D6**	TIMER_A_PWM1	PE9	TIM1_CH1
5	**GND**	GND	-	GND	6	**D5**	TIMER_A_PWM2	PE11	TIM1_CH2
7	**A6**	ADC_A_IN	PF4	ADC3_INP9	8	**D4**	I/O	PE14	I/O
9	**A7**	ADC_B_IN	PF5	ADC3_INP4	10	**D3**	TIMER_A_PWM3	PE13	TIM1_CH3
11	**A8**	ADC_C_IN	PF6	ADC3_INP8	12	**D2**	I/O	PG14	I/O
13	**D26**	QSPI_CS	PG6	QSPI1_NCS	14	**D1**	USART_A_TX	PB6	LPUART1
15	**D27**	QSPI_CLK	PB2	QSPI1_CLK	16	**D0**	USART_A_RX	PB7	LPUART1
17	**GND**	GND	-	GND	18	**D42**	TIMER_A_PWM1N	PE8	TIM1_CH1N
19	**D28**	QSPI_BK1_IO3	PD13	QSPI1_IO	20	**D41**	TIMER_A_ETR	PE7	TIM1_ETR
21	**D29**	QSPI_BK1_IO1	PD12	QSPI1_IO	22	**GND**	GND	-	GND
23	**D30**	QSPI_BK1_IO0	PD11	QSPI1_IO	24	**D40**	TIMER_A_PWM2N	PE10	TIM1_CH2N
25	**D31**	QSPI_BK1_IO2	PE2(1)	QSPI1_IO	26	**D39**	TIMER_A_PWM3N	PE12	TIM1_CH3N
27	**GND**	-	-	-	28	**D38**	TIMER_A_BKIN2	PE6	TIM1_BKIN2
29	**D32**	TIM_C_PWM1	PA0	TIM2_CH1	30	**D37**	TIMER_A_BKIN1	PE15	TIM1_BKIN1
31	**D33**	TIM_D_PWM1	PB0	TIM3_CH3	32	**D36**	TIMER_C_PWM2	PB10	TIM2_CH3
33	**D34**	TIM_B_ETR	PE0	TIM4_ETR	34	**D35**	TIMER_C_PWM3	PB11	TIM2_CH4

8.2 ST morpho connector

The ST morpho connector consists in male pin header footprints CN11 and CN12 (not soldered by default). They are used to connect the STM32H7 Nucleo-144 board to an extension board or a prototype/wrapping board placed on top of the STM32H7 Nucleo-144 board. All signals and power pins of the STM32H7 are available on the ST morpho connector. This connector can also be probed by an oscilloscope, logical analyzer or voltmeter.

Table 22 shows the pin assignments of each STM32H7 on the ST morpho connector.

Table 22. ST morpho connector pin assignment

CN11 odd pins		CN11 even pins		CN12 odd pins		CN12 even pins	
Pin nbr	Pin name	Pin nbr	Pin name	Pin nbr	Pin name	Pin nbr	Pin name
1	PC10	**2**	PC11	**1**	PC9	**2**	PC8
3	PC12	**4**	PD2	**3**	PB8	**4**	PC6
5	3V3_VDD	**6**	5V_EXT	**5**	PB9	**6**	PC5
7	BOOT0(1)	**8**	GND	**7**	VREFP	**8**	5V_USB_STLK(2)
9	PF6	**10**	NC	**9**	GND	**10**	PD8
11	PF7	**12**	IOREF	**11**	PA5	**12**	PA12
13	PA13(3)	**14**	NRST	**13**	PA6	**14**	PA11
15	PA14(3)	**16**	3V3	**15**	PA7	**16**	PB12
17	PA15	**18**	5V	**17**	PB6	**18**	PB11
19	GND	**20**	GND	**19**	PC7	**20**	GND
21	PB7	**22**	GND	**21**	PA9	**22**	PB2
23	PC13	**24**	VIN	**23**	PA8	**24**	PB1
25	PC14	**26**	NC	**25**	PB10	**26**	PB15
27	PC15	**28**	PA0	**27**	PB4	**28**	PB14
29	PH0	**30**	PA1	**29**	PB5	**30**	PB13
31	PH1	**32**	PA4	**31**	PB3	**32**	AGND
33	VBAT	**34**	PB0	**33**	PA10	**34**	PC4
35	PC2	**36**	PC1	**35**	PA2	**36**	PF5
37	PC3	**38**	PC0	**37**	PA3	**38**	PF4
39	PD4	**40**	PD3	**39**	GND	**40**	PE8
41	PD5	**42**	PG2	**41**	PD13	**42**	PF10
43	PD6	**44**	PG3	**43**	PD12	**44**	PE7
45	PD7	**46**	PE2	**45**	PD11	**46**	PD14
47	PE3	**48**	PE4	**47**	PE10	**48**	PD15
49	GND	**50**	PE5	**49**	PE12	**50**	PF14
51	PF1	**52**	PF2	**51**	PE14	**52**	PE9

Table 22. ST morpho connector pin assignment (continued)

CN11 odd pins		CN11 even pins		CN12 odd pins		CN12 even pins	
Pin nbr	Pin name	Pin nbr	Pin name	Pin nbr	Pin name	Pin nbr	Pin name
53	PF0	**54**	PF8	**53**	PE15	**54**	GND
55	PD1	**56**	PF9	**55**	PE13	**56**	PE11
57	PD0	**58**	PG1	**57**	PF13	**58**	PF3
59	PG0	**60**	GND	**59**	PF12	**60**	PF15
61	PE1	**62**	PE6	**61**	PG14	**62**	PF11
63	PG9	**64**	PG15	**63**	GND	**64**	PE0
65	PG12	**66**	PG10	**65**	PD10	**66**	PG8
67	NC	**68**	PG13	**67**	PG7	**68**	PG5
69	PD9	**70**	PG11	**69**	PG4	**70**	PG6

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is plugged on the pins 5-7 of CN11.
2. 5V_USB_STLK is the 5 V power coming from the ST-LINKV3 USB connector that rises before and it rises before the +5 V rising on the board.
3. PA13 and PA14 are shared with SWD signals connected to STLINK-V3. It is not recommended to use them as I/O pins.

Appendix A Federal Communications Commission (FCC) and Industry Canada (IC) Compliance

A.1 FCC Compliance Statement

A.1.1 Part 15.19

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

A.1.2 Part 15.105

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

A.1.3 Part 15.21

Any changes or modifications to this equipment not expressly approved by STMicroelectronics may cause harmful interference and void the user's authority to operate this equipment.

A.2 IC Compliance Statement

This device complies with FCC and Industry Canada RF radiation exposure limits set forth for general population for mobile application (uncontrolled exposure). This device must not be collocated or operating in conjunction with any other antenna or transmitter.

A.2.1 Compliance Statement

Notice: This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Industry Canada ICES-003 Compliance Label: CAN ICES-3 (A)/NMB-3(A)

A.2.2 Déclaration de conformité

Avis: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit

accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement

Étiquette de conformité à la NMB-003 d'Industrie Canada: CAN ICES-3 (A)/NMB-3(A)

Appendix B CISPR32

B.1 Warning

Warning: This device is compliant with Class A of CISPR32. In a residential environment, this equipment may cause radio interference.

Avertissement: Cet équipement est conforme à la Classe A de la CISPR 32. Dans un environnement résidentiel, cet équipement peut créer des interférences radio.

Revision history

Table 23. Document revision history

Date	Revision	Changes
14-Mar-2019	1	Initial version

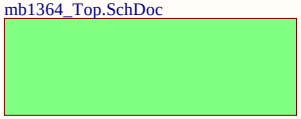
NUCLEO 144 H7 LEGACY

MB1364

H743ZI

Table of contents

U_mb1364_Top
mb1364_Top.SchDoc



Legend

General comment such as function title, configuration, ...

Text to be added to silkscreen.

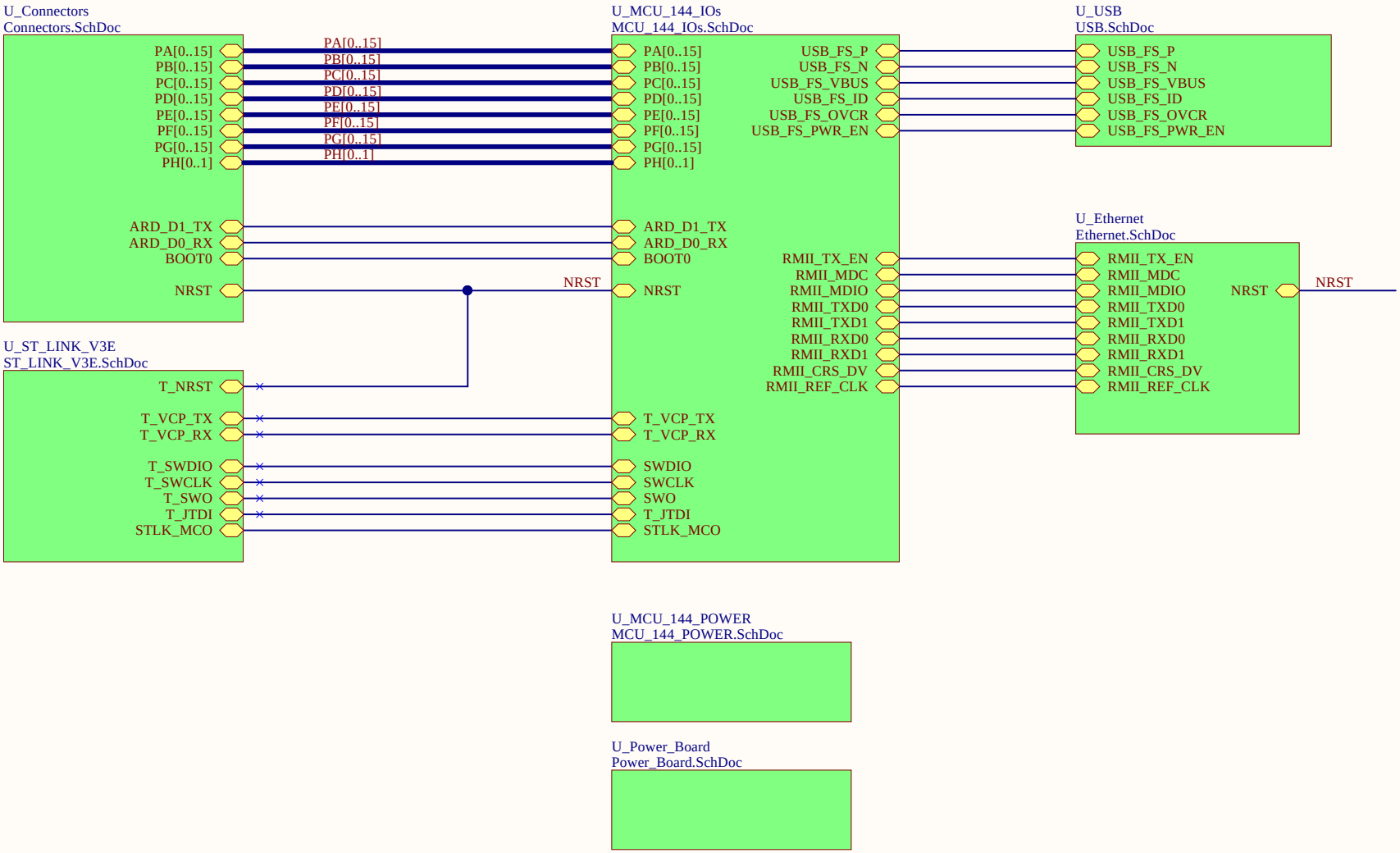
Warning text.

Notes to generate the board layout.

OPEN PLATFORM LICENSE AGREEMENT

Title: **Project overview**		
Project: **NUCLEO 144 H7 LEGACY**		
Variant: H743ZI		
Revision: C-01		Reference: MB1364
Size: A4	Date: 17-DEC-2018	Sheet: 1 of 9

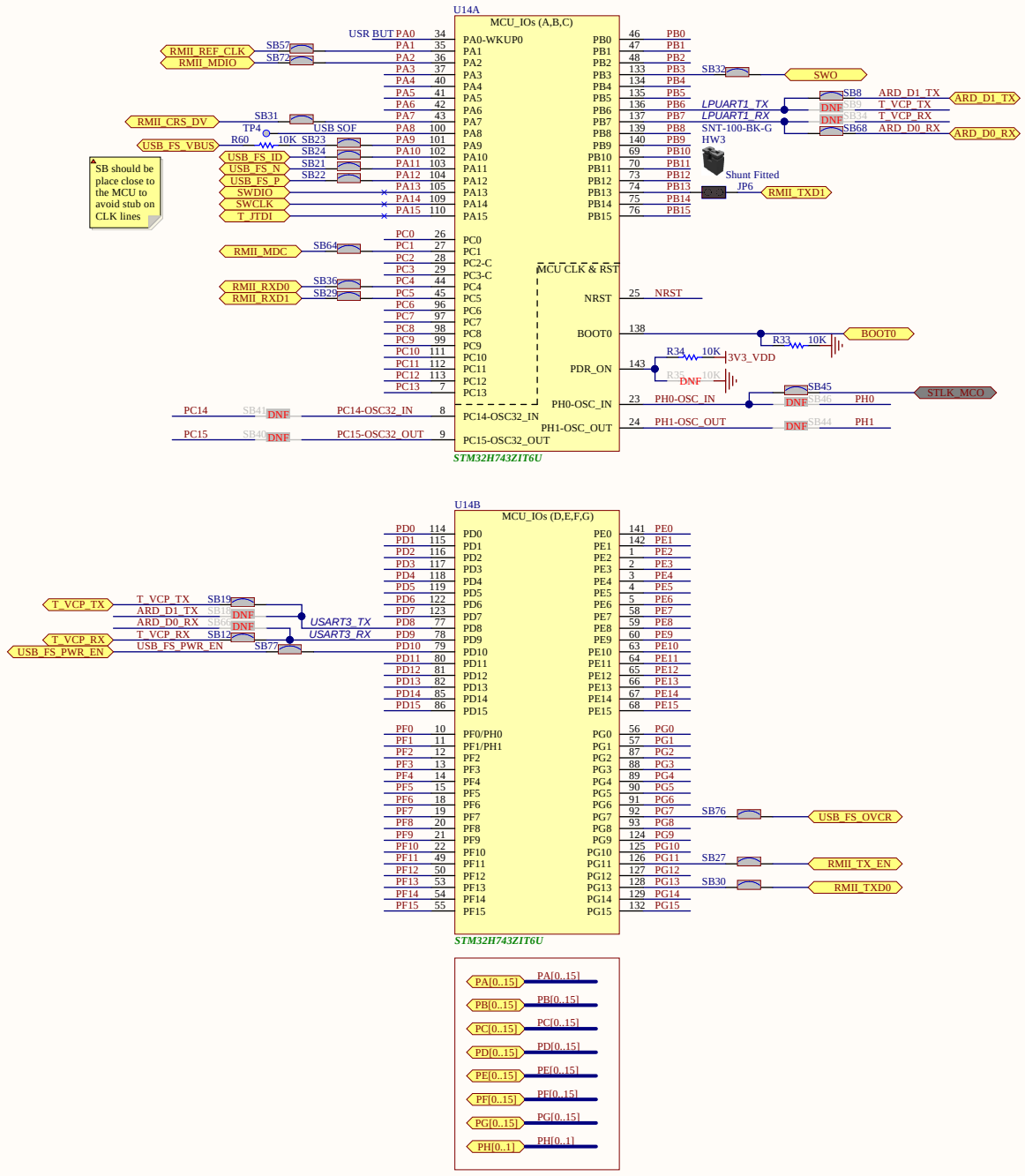
life.augmented



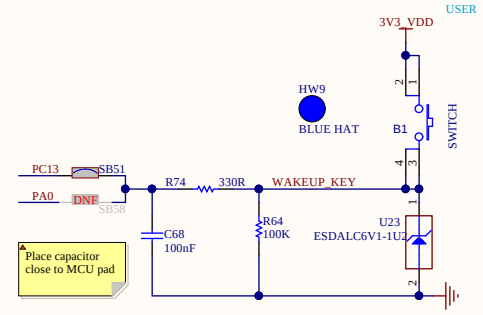
Title: **mb1319 Top: hierarchical view**		
Project: **NUCLEO 144 H7 LEGACY**		
Variant: H743ZI		
Revision: C-01		Reference: MB1364
Size: A4	Date: 17-DEC-2018	Sheet: 2 of 9

life.augmented

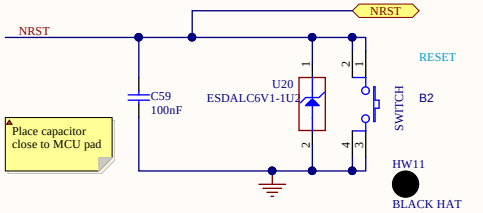
MCU IO CLK and RST



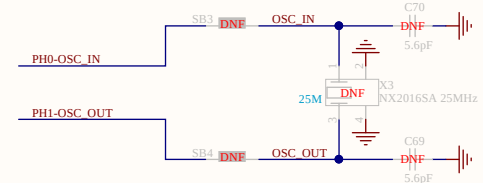
USER BUTTON



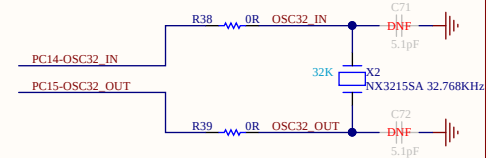
RESET FUNCTION



EXTERNAL HSE CLK
(OPTIONAL)

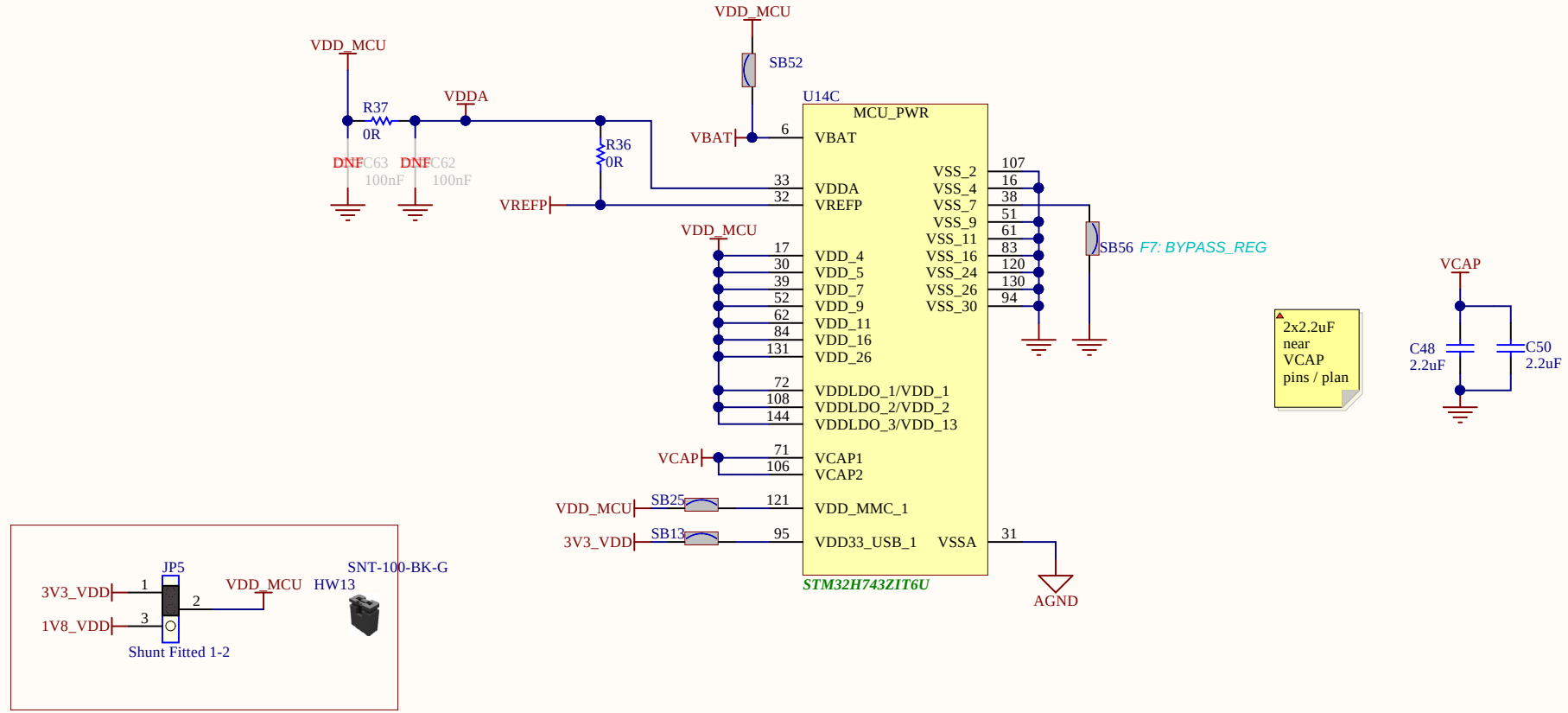


EXTERNAL LSE CLK

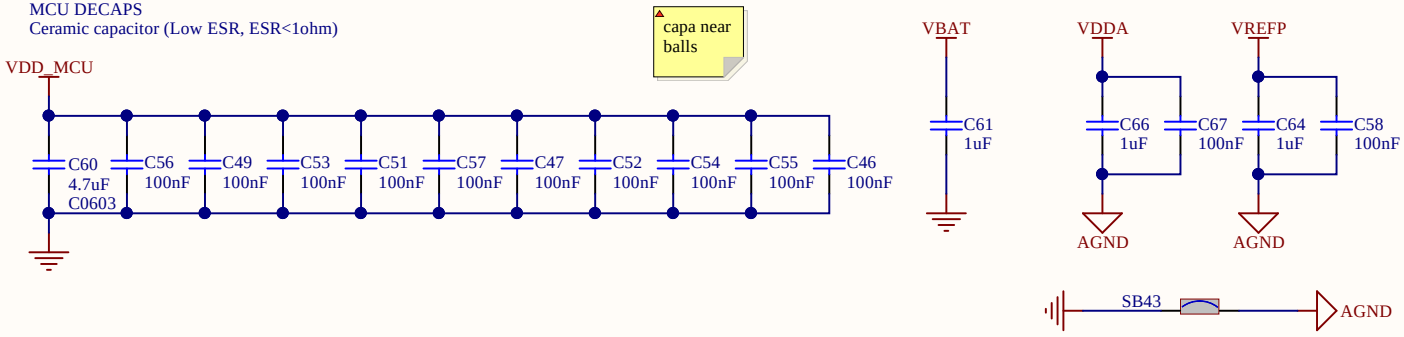


Title: **MCU 144 I/Os**	
Project: **NUCLEO 144 H7 LEGACY**	
Variant: H743ZI	
Revision: C-01	Reference: MB1364
Size: A3	Date: 17-DEC-2018
	Sheet: 3 of 9

MCU PWR SUPPLIES

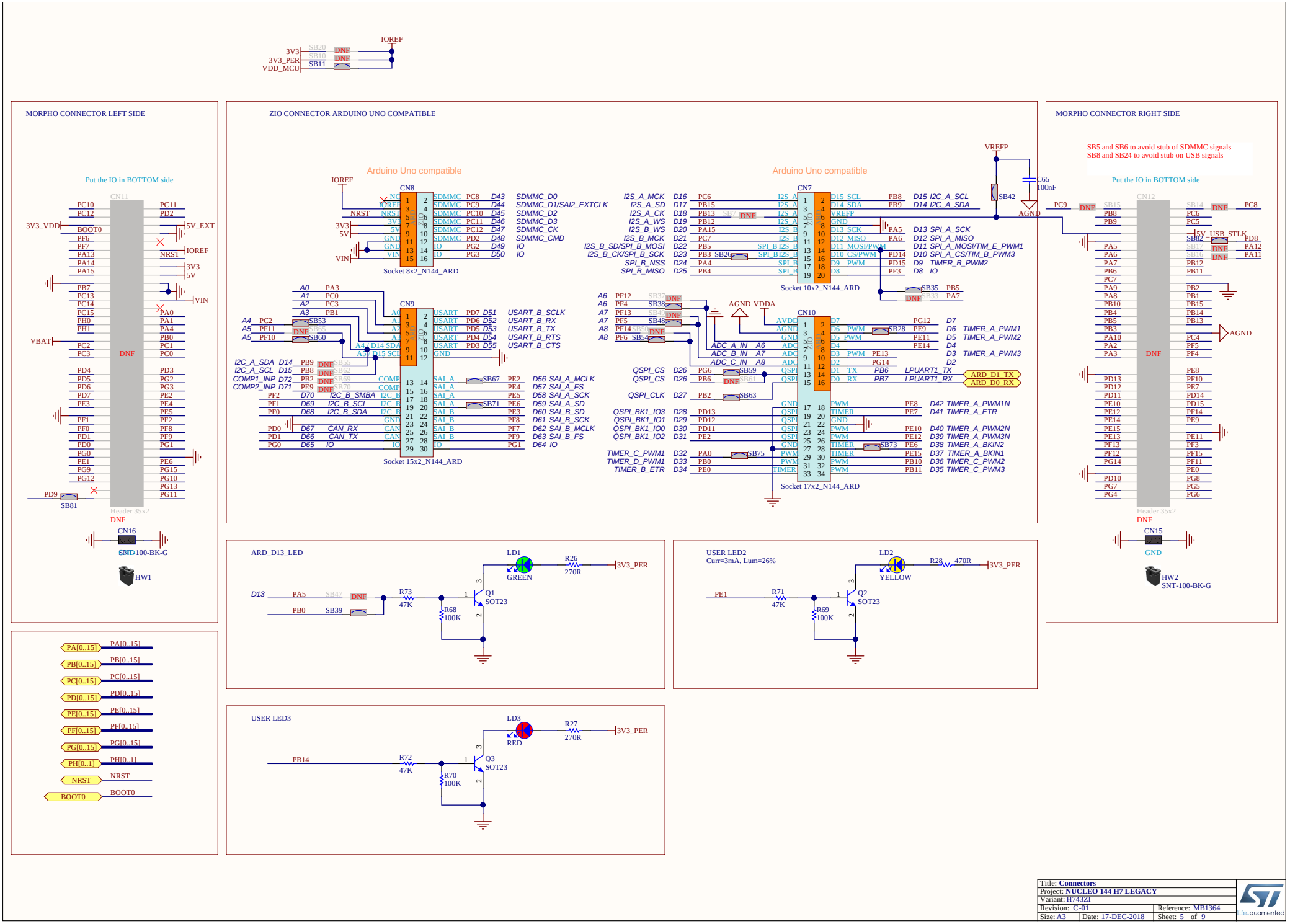


MCU DECAPS
Ceramic capacitor (Low ESR, ESR<1ohm)

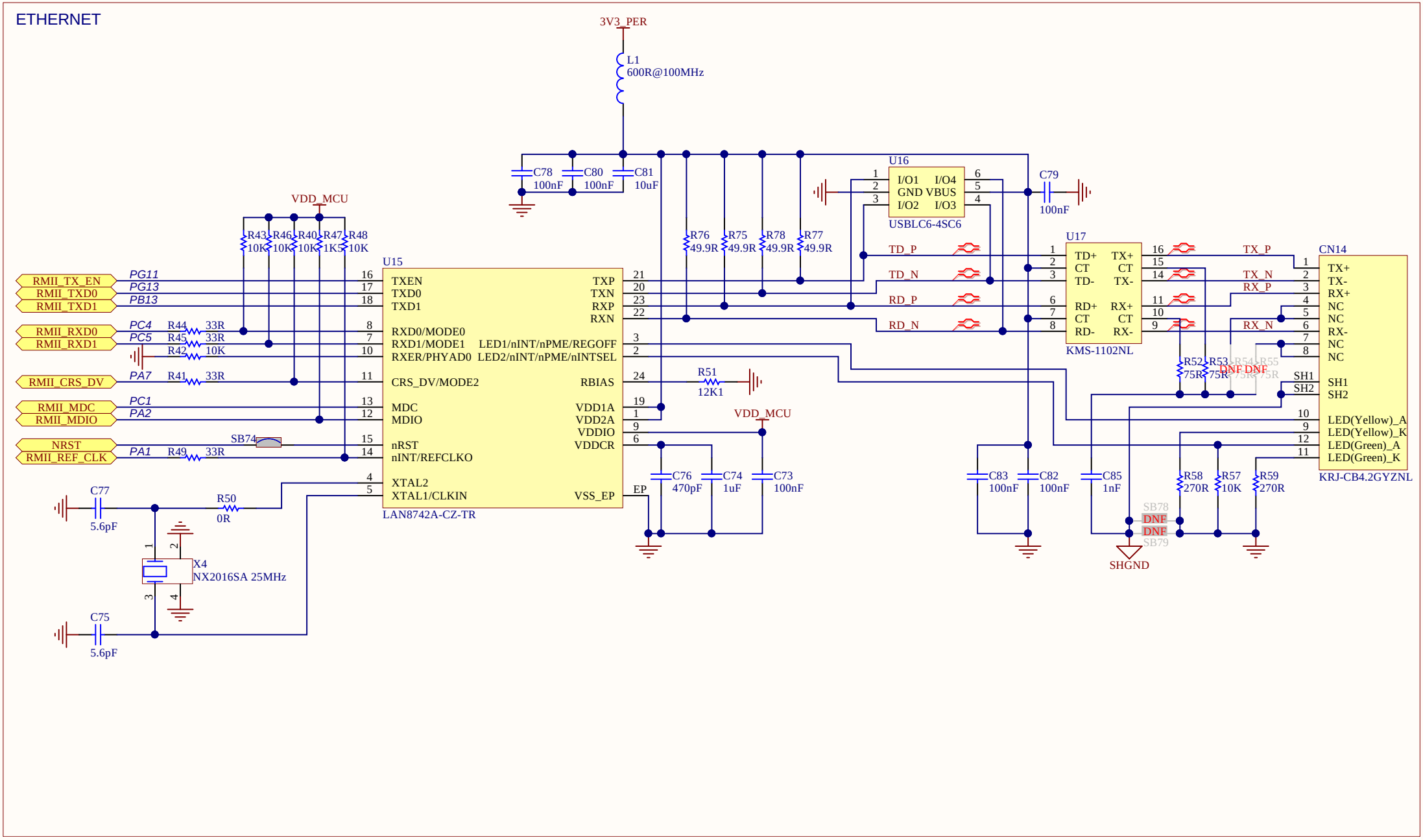


Title: **MCU 144 POWER**		
Project: **NUCLEO 144 H7 LEGACY**		
Variant: H743ZI		
Revision: C-01		Reference: MB1364
Size: A4	Date: 17-DEC-2018	Sheet: 4 of 9

life.augmented



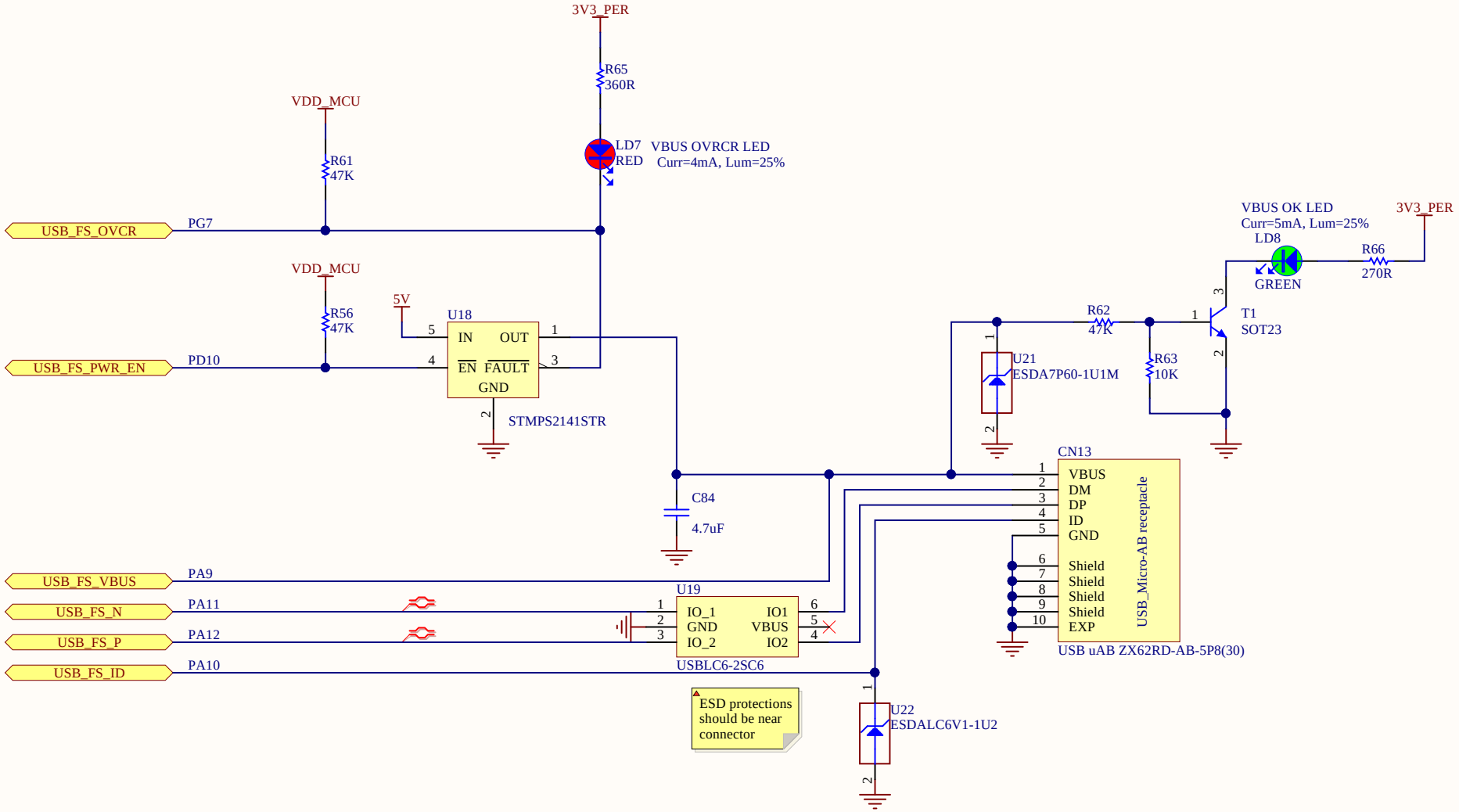
Title: **Connectors**			
Project: **NUCLEO 144 H7 LEGACY**			
Variant: H7432I			
Revision: C-01		Reference: MB1364	
Size: A3	Date: 17-DEC-2018	Sheet: 5 of 9	



Title: **Ethernet PHY with RJ45 connector**		
Project: **NUCLEO 144 H7 LEGACY**		
Variant: H743ZI		
Revision: C-01		Reference: MB1364
Size: A4	Date: 17-DEC-2018	Sheet: 6 of 9

life.augmented

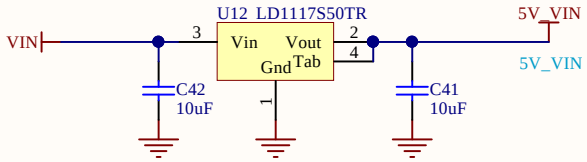
USB



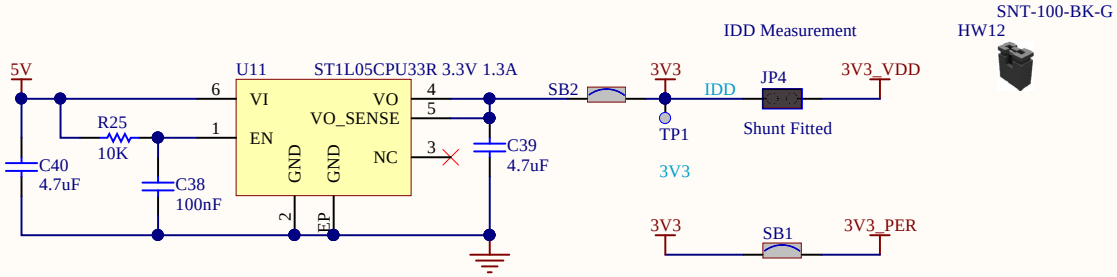
Title: **USB**		
Project: **NUCLEO 144 H7 LEGACY**		
Variant: H743ZI		
Revision: C-01		Reference: MB1364
Size: A4	Date: 17-DEC-2018	Sheet: 7 of 9

life.augmented

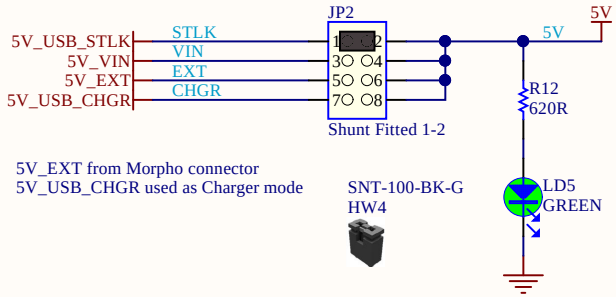
VIN / 5V PWR



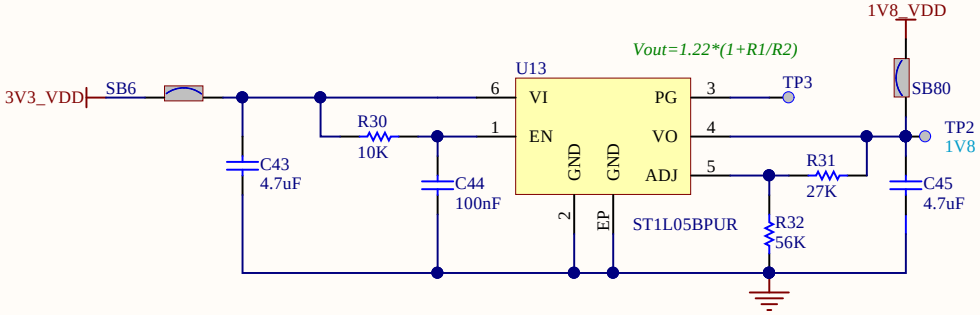
3V3 PWR



5V PWR SELECTION

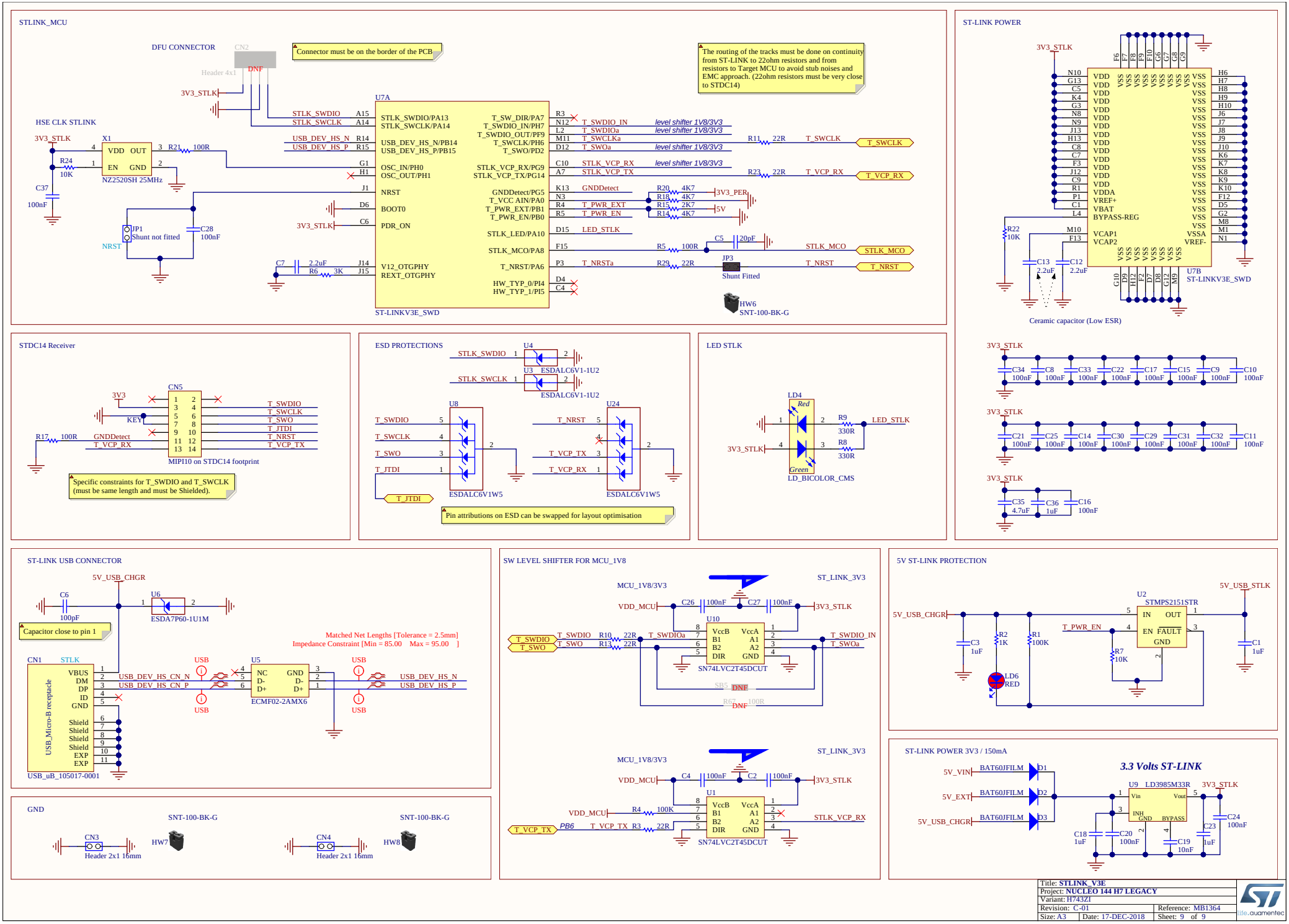


1V8 PWR



Title: **MAIN POWER 5V / 3V3 / 1V8**		
Project: **NUCLEO 144 H7 LEGACY**		
Variant: H743ZI		
Revision: C-01		Reference: MB1364
Size: A4	Date: 17-DEC-2018	Sheet: 8 of 9

life.augmented



Title:	**STLINK_V3E**
Project:	**NUCLEO 144 H7 LEGACY**
Variant:	H7432I
Revision:	C-01
Reference:	MB1364
Size:	A3
Date:	17-DEC-2018
Sheet:	9 of 9