Comparison between **STM32L152RCT6** and **STM32L476VGT6**

	STM32L1	**STM32L4**
Core	Cortex-M**3** @ 32MHz with **64** pins	Cortex-M**4** @ 80MHz with **FPU and DSP** with **100** pins
MSI	64 kHz, 128 kHz, 256 kHz, 512 kHz, 1.02 MHz, **2.05 MHz (default value)**, 4.1 MHz	100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz, 2 MHz, **4 MHz (default value)**, 8 MHz, 16 MHz, 24 MHz, 32 MHz and 48 MHz
LSI	37 kHz	32 kHz RC
HSE	1 – 24 MHz	4 – 48 MHz
System clock	Up to 32 MHz Default to **MSI 2MHz** after reset	Up to 80 MHz Default to **MSI 4MHz** after reset
RCC	RCC_AHBENR	RCC_AHB**1**ENR (AHB1) RCC_AHB**2**ENR (AHB2) RCC_AHB**3**ENR (AHB3)
	RCC_AHB**LP**ENR (LP = Low Power)	RCC_AHB**1SM**ENR (AHB1) RCC_AHB**2SM**ENR (AHB2) RCC_AHB**3SM**ENR (AHB3) (SM = Sleep Mode)
	RCC_APB1ENR	RCC_APB1ENR**1** RCC_APB1ENR**2**
	RCC_APB1**LP**ENR (LP = Low Power)	RCC_APB1**SM**ENR**1** RCC_APB1**SM**ENR**2** (SM = Sleep Mode)
	RCC_APB2**LP**ENR (LP = Low Power)	RCC_APB2**SM**ENR (SM = Sleep Mode)
	The MSIRANGE in **RCC_ICSCR** selects the MSI frequency.	The MSIRANGE in **RCC_CR** or **RCC_CSR** selects the MSI frequency. The MSIRGSEL bit in RCC_CR determines which MSIRANGE is used. • If MSIRGSEL is 0 (default), the MSIRANGE in RCC_CSR is used to select the MSI clock range. • If MSIRGSEL is 1, the MSIRANGE in RCC_CR is used.

<table>
<tr><th></th><th>STM32L1</th><th>STM32L4</th></tr>
<tr><td rowspan="2">GPIO</td><td>Default mode is Digital Input</td><td>Default mode is Analog</td></tr>
<tr><td>Alternative functions are different.

AF0 SYSTEM
AF1 TIM2
AF2 TIM3/TIM4/TIM5
AF3 TIM9/TIM10/TIM11
AF4 I2C1/I2C2
AF5 SPI1/SPI2
AF6 SPI3
AF7 USART1/ USART2/ USART3
AF8 UART4/UART5
AF9
AF10 USB
AF11 LCD
AF12 FSMC
AF13
AF14 RI
AF15 EVENTOUT</td><td>Alternative functions are different.

AF0 SYSTEM
AF1 TIM1/TIM2/TIM5/TIM8/LPTIM1
AF2 TIM1/TIM2/TIM3/TIM4/TIM5
AF3 TIM8
AF4 I2C1/I2C2/I2C3
AF5 SPI1/SPI2
AF6 SPI3/DFSDM
AF7 USART1/USART2/ USART3
AF8 UART4/UART5/LPUART1
AF9 CAN1/TSC
AF10 OTG_FS/QUADSPI
AF11 LCD
AF12 SDMMC1/COMP1/COMP2/FMC/SWPMI1
AF13 SAI1/SAI2
AF14 TIM2/TIM15/TIM16/TIM17/LPTIM2
AF15 EVENTOUT</td></tr>
<tr><td></td><td>Add a new register GPIO_ASCR (Analog Switch Control Register)

0: Disconnect analog switch to the ADC input
1: Connect analog switch to the ADC input

<pre>typedef struct {
  __IO uint32_t MODER;
  __IO uint32_t OTYPER;
  __IO uint32_t OSPEEDR;
  __IO uint32_t PUPDR;
  __IO uint32_t IDR;
  __IO uint32_t ODR;
  __IO uint32_t BSRR;
  __IO uint32_t LCKR;
  __IO uint32_t AFR[2];
  __IO uint32_t BRR;
  __IO uint32_t ASCR;
} GPIO_TypeDef;</pre>
For example, to use PA.2 as analog ADC input:
GPIOA->ASCR |= 1U<<2;</td></tr>
</table>

	STM32L1	**STM32L4**
RTC	typedef struct { … __IO uint32_t **CALIBR**; … uint32_t RESERVED7; … } RTC_TypeDef;	typedef struct { … uint32_t reserved; … __IO uint32_t **OR**; … } RTC_TypeDef;
	Digital calibration (DC) is configured through bits DC[4:0] of RTC_CALIBR register. This number ranges from 0 to 31 corresponding to a time interval (2xDC) ranging from 0 to 62.	RTC_CALIBR register is not available
	RTC option register (RTC_OR) is not available.	RTC_OR controls • remap of the RTC outputs (RTC_OUT) on PB2 • RTC_ALARM on PC13 output type
	The Calibration output (COE) bit in RTC_CR enables the RTC_CALIB output.	The COE bit is not available in RTC_CR.
	RTC tamper and alternate function configuration register (RTC_TAFCR)	RTC tamper configuration register (RTC_TAMPCR). Bit ALARMOUTTYPE available in RTC_OR register

	STM32L1	STM32L4
EXTI	Up to 24 lines	Up to 40 lines (14 direct, 26 configurable)
	typedef struct{ __IO uint32_t IMR; __IO uint32_t EMR; __IO uint32_t RTSR; __IO uint32_t FTSR; __IO uint32_t SWIER; __IO uint32_t PR; } EXTI_TypeDef;	typedef struct{ __IO uint32_t IMR**1**; __IO uint32_t EMR**1**; __IO uint32_t RTSR**1**; __IO uint32_t FTSR**1**; __IO uint32_t SWIER**1**; __IO uint32_t PR**1**; uint32_t RESERVED1; uint32_t RESERVED2; __IO uint32_t IMR**2**; __IO uint32_t EMR**2**; __IO uint32_t RTSR**2**; __IO uint32_t FTSR**2**; __IO uint32_t SWIER**2**; __IO uint32_t PR**2**; } EXTI_TypeDef;
	The selection of EXTI line source is performed through EXTIx bits in SYSCFG_EXTICRx registers (in STM32L1 and STM32L4 series). SYSCFG_EXTICR1 SYSCFG_EXTICR2 SYSCFG_EXTICR3 SYSCFG_EXTICR4 EXTIx[**3**:0]: 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin 0101: PH[x] (only PH[2:0]) PH[3] is not used.	The mapping of the EXTICRx registers has been changed. SYSCFG_EXTICR1 SYSCFG_EXTICR2 SYSCFG_EXTICR3 SYSCFG_EXTICR4 EXTIx[**2**:0]: 000: PA[x] pin 001: PB[x] pin 010: PC[x] pin 011: PD[x] pin 100: PE[x] pin 101: PF[x] pin 110: PG[x] pin 111: Reserved

	STM32L1	**STM32L4**
USART	3 USART, 2 UART	3 USART, 2 UART, 1 LPUART
	up to 4 Mbit/s (when the clock frequency is 32 MHz and oversampling is by 8)	up to 10 Mbit/s (when the clock frequency is 80 MHz and oversampling is by 8)
	Programmable word length (8 or 9 bits)	Programmable word length (7, 8 or 9 bits), programmable data order with MSB-first or LSB-first shifting
	10 interrupt sources with flags	14 interrupt sources with flags
	U(S)ART clock is APB1 or APB2 clock	U(S)ART clock is derived from one of the four following sources: system clock (SYSCLK), HSI16, LSE, APB1 or APB2 clock

	Data structure
	`typedef struct {`
	`  __IO uint16_t SR;`
	`  uint16_t  RESERVED0;`
	`  __IO uint16_t DR;`
	`  uint16_t  RESERVED1;`
	`  __IO uint16_t BRR;`
	`  uint16_t  RESERVED2;`
	`  __IO uint16_t CR1;`
	`  uint16_t  RESERVED3;`
	`  __IO uint16_t CR2;`
	`  uint16_t  RESERVED4;`
	`  __IO uint16_t CR3;`
	`  uint16_t  RESERVED5;`
	`  __IO uint16_t GTPR;`
	`  uint16_t  RESERVED6;`
	`} USART_TypeDef;`
		`typedef struct {`
		`  __IO uint32_t CR1;`
		`  __IO uint32_t CR2;`
		`  __IO uint32_t CR3;`
		`  __IO uint32_t BRR;`
		`  __IO uint16_t GTPR;`
		`  uint16_t  RESERVED2;`
		`  __IO uint32_t RTOR;`
		`  __IO uint16_t RQR;`
		`  uint16_t  RESERVED3;`
		`  __IO uint32_t ISR;`
		`  __IO uint32_t ICR;`
		`  __IO uint16_t RDR;`
		`  uint16_t  RESERVED4;`
		`  __IO uint16_t TDR;`
		`  uint16_t  RESERVED5;`
		`} USART_TypeDef;`

	`USARTx->DR`	`USARTx->TDR or USARTx->RDR`
	`USARTx->SR`	`USARTx->ISR`
	`Clear status flags via USARTx->SR`	`Clear status flags via USARTx->ICR`

	STM32L1	**STM32L4**
I2C	I2C1, I2C2	I2C1, I2C2, I2C3
	Standard mode (up to 100 kHz), Fast mode (up to 400 kHz)	Standard mode (up to 100 kHz), Fast mode (up to 400 kHz), Fast mode Plus (up to 1 MHz)
	I2C clock is APB1 clock (PCLK1).	I2C clock is derived from one of the three following sources: system clock (SYSCLK), HSI16, APB1 (PCLK1).

	`typedef struct {`
	`  __IO uint16_t CR1;`
	`  __IO uint16_t CR2;`
	`  __IO uint16_t OAR1;`
	`  __IO uint16_t OAR2;`
	`  __IO uint16_t DR;`
	`  __IO uint16_t SR1;`
	`  __IO uint16_t SR2;`
	`  __IO uint16_t CCR;`
	`  __IO uint16_t TRISE;`
	`} I2C_TypeDef;`
		`typedef struct {`
		`  __IO uint32_t CR1;`
		`  __IO uint32_t CR2;`
		`  __IO uint32_t OAR1;`
		`  __IO uint32_t OAR2;`
		`  __IO uint32_t TIMINGR;`
		`  __IO uint32_t TIMEOUTR;`
		`  __IO uint32_t ISR;`
		`  __IO uint32_t ICR;`
		`  __IO uint32_t PECR;`
		`  __IO uint32_t RXDR;`
		`  __IO uint32_t TXDR;`
		`} I2C_TypeDef;`

	`I2C->DR`	`I2C->RXDR or I2C->TXDR`

	STM32L1	STM32L4
SPI	Data size is fixed, configurable to 8 or 16 bits	Data size is programmable, from 4 to 16-bit
	Tx & Rx 16-bit buffers (single data frame)	32-bit Tx & Rx FIFOs (up to 4 data frames) Rx buffer not empty (RXNE): The RXNE flag is set depending on the FRXTH bit value in the SPIx_CR2 register: (1) If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit). (2) If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit). **SPIx->CR2 |= SPI_CR2_FRXTH;**
	No data packing (16-bit access only) `SPIx->DR = byte_data;`	Data packing (8-bit, 16-bit or 32-bit data access, programmable FIFOs data thresholds) **`*((volatile uint8_t*)&SPIx->DR) = byte_data;`** **`byte_data = (uint8_t)(SPIx->DR);`**
		The data size and Tx/Rx flow handling are different in STM32L1 and STM32L4 series hence requiring different SW sequences

	`typedef struct{`
	`__IO uint16_t CR1;`
	`__IO uint16_t CR2;`
	`__IO uint16_t SR;`
	`__IO uint16_t DR;`
	`__IO uint16_t CRCPR;`
	`__IO uint16_t RXCRCR;`
	`__IO uint16_t TXCRCR;`
	`} SPI_TypeDef;`
		`typedef struct{`
		`__IO uint32_t CR1;`
		`__IO uint32_t CR2;`
		`__IO uint32_t SR;`
		`__IO uint32_t DR;`
		`__IO uint32_t CRCPR;`
		`__IO uint32_t RXCRCR;`
		`__IO uint32_t TXCRCR;`
		`} SPI_TypeDef;`

	STM32L1	**STM32L4**
ADC	ADC1	ADC1, ADC2, ADC3
	Max speed: 1 Msps	Max speed: 5.1 Msps (fast channel), 4.8 Msps (slow channel)
	12-bit	12-bit + digital oversampling up to 16-bit
	Reference Voltage: external	Reference Voltage: external (2.0 V to VDDA) or internal (2.048 V or 2.5 V)
	ADCCLK is always the HSI oscillator clock.	The ADCs clock can be derived (selected by software) from one of the three following sources: • system clock (SYSCLK), • PLLSAI1 VCO (PLLADC1CLK), • PLLSAI2 VCO (PLLADC2CLK).

	typedef struct {
	__IO uint32_t SR;
	__IO uint32_t CR1;
	__IO uint32_t CR2;
	__IO uint32_t SMPR1;
	__IO uint32_t SMPR2;
	__IO uint32_t SMPR3;
	__IO uint32_t JOFR1;
	__IO uint32_t JOFR2;
	__IO uint32_t JOFR3;
	__IO uint32_t JOFR4;
	__IO uint32_t HTR;
	__IO uint32_t LTR;
	__IO uint32_t SQR1;
	__IO uint32_t SQR2;
	__IO uint32_t SQR3;
	__IO uint32_t SQR4;
	__IO uint32_t SQR5;
	__IO uint32_t JSQR;
	__IO uint32_t JDR1;
	__IO uint32_t JDR2;
	__IO uint32_t JDR3;
	__IO uint32_t JDR4;
	__IO uint32_t DR;
	} ADC_TypeDef;
		typedef struct{
		__IO uint32_t ISR;
		__IO uint32_t IER;
		__IO uint32_t CR;
		__IO uint32_t CFGR;
		__IO uint32_t CFGR2;
		__IO uint32_t SMPR1;
		__IO uint32_t SMPR2;
		__IO uint32_t TR1;
		__IO uint32_t TR2;
		__IO uint32_t TR3;
		__IO uint32_t SQR1;
		__IO uint32_t SQR2;
		__IO uint32_t SQR3;
		__IO uint32_t SQR4;
		__IO uint32_t DR;
		__IO uint32_t JSQR;
		__IO uint32_t OFR1;
		__IO uint32_t OFR2;
		__IO uint32_t OFR3;
		__IO uint32_t OFR4;
		__IO uint32_t JDR1;
		__IO uint32_t JDR2;
		__IO uint32_t JDR3;
		__IO uint32_t JDR4;
		__IO uint32_t AWD2CR;
		__IO uint32_t AWD3CR;
		__IO uint32_t DIFSEL;
		__IO uint32_t CALFACT;
		} ADC_TypeDef;
		
		typedef struct {
		__IO uint32_t CSR;
		__IO uint32_t CCR;
		__IO uint32_t CDR;
		} ADC_Common_TypeDef;

	STM32L1	STM32L4
DAC	External trigger: • TIM6 TRGO • TIM7 TRGO • **TIM9 TRGO** • TIM2 TRGO • TIM4 TRGO • EXTI line9 • SW TRIG	External trigger: • TIM6 TRGO • **TIM8 TRGO** • TIM7 TRGO • **TIM5 TRGO** • TIM2 TRGO • TIM4 TRGO • EXTI line9 • SW TRIG
	typedef struct { __IO uint32_t CR; __IO uint32_t SWTRIGR; __IO uint32_t DHR12R1; __IO uint32_t DHR12L1; __IO uint32_t DHR8R1; __IO uint32_t DHR12R2; __IO uint32_t DHR12L2; __IO uint32_t DHR8R2; __IO uint32_t DHR12RD; __IO uint32_t DHR12LD; __IO uint32_t DHR8RD; __IO uint32_t DOR1; __IO uint32_t DOR2; __IO uint32_t SR; } DAC_TypeDef;	typedef struct { __IO uint32_t CR; __IO uint32_t SWTRIGR; __IO uint32_t DHR12R1; __IO uint32_t DHR12L1; __IO uint32_t DHR8R1; __IO uint32_t DHR12R2; __IO uint32_t DHR12L2; __IO uint32_t DHR8R2; __IO uint32_t DHR12RD; __IO uint32_t DHR12LD; __IO uint32_t DHR8RD; __IO uint32_t DOR1; __IO uint32_t DOR2; __IO uint32_t SR; __IO uint32_t **CCR**; __IO uint32_t **MCR**; __IO uint32_t **SHSR1**; __IO uint32_t **SHSR2**; __IO uint32_t **SHHR**; __IO uint32_t **SHRR**; } DAC_TypeDef;
	Reference Voltage: • External (2.0 V to VDDA, or 1.8 V to VDDA)	Reference Voltage: • external (2.0 V to VDDA) • internal (2.048 V or 2.5 V)