STM32 HAL LIBRARY CHEAT SHEET

LIBRARY:

```
#include "stm32f0xx_hal.h"
```

DIGITAL INPUT:

```
HAL_GPIO_ReadPin ( GPIOX, GPIO_PIN_X)
```

DIGITAL OUTPUT:

```
HAL_GPIO_WritePin ( GPIOX, GPIO_PIN_X, GPIO_PIN_SET)
HAL_GPIO_WritePin ( GPIOX, GPIO_PIN_X, GPIO_PIN_RESET)
HAL_GPIO_TogglePin ( GPIOX, GPIO_PIN_X)
```

ANALOG INPUT:

```
ADC_HandleTypeDef hadcX // Global var

HAL_ADC_Start ( &hadcX )
HAL_ADC_PollForConversion ( &hadcX, TIMEOUT_MS )
uint32_t value_adc = HAL_ADC_GetValue ( &hadcX )
HAL_ADC_Stop ( &hadcX )
```

CONTROL FUNCTIONS:

```
HAL_Delay( TIMEOUT_MS )
```

INIT DIGITAL INPUT PIN:

```
// Enable port clock
__HAL_RCC_GPIOX_CLK_ENABLE ( );

GPIO_InitStruct.Pin = GPIO_PIN_X;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL /_PULLUP _PULLDOWN
 HAL_GPIO_Init ( GPIOX, &GPIO_InitStruct);
```

INIT DIGITAL OUTPUT PIN:

```
// Enable port clock
__HAL_RCC_GPIOX_CLK_ENABLE ( );

GPIO_InitStruct.Pin = GPIO_PIN_X
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
GPIO_InitStruct.Pull = GPIO_NOPULL
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW
HAL_GPIO_Init ( GPIOX, &GPIO_InitStruct )
```

INIT ANALOG INPUT PIN:

```
// Enable port clock
__HAL_RCC_GPIOX_CLK_ENABLE ( );

GPIO_InitStruct.Pin = GPIO_PIN_X
GPIO_InitStruct.Mode = GPIO_MODE_ANALOG
GPIO_InitStruct.Pull = GPIO_NOPULL
 HAL_GPIO_Init ( GPIOX, &GPIO_InitStruct )
```

STM32 HAL LIBRARY CHEAT SHEET

TIMER (Counts up to COUNT_MAX and then resets)

```
// INIT THE TIMER
TIM_HandleTypeDef htim1 // Global var

void MX_TIM2_Init(void) {
   htim1.Instance = TIMX;
   htim1.Init.Prescaler = PREESCALER_VAL // F_tim = F_clock/PRE-1
   htim1.Init.CounterMode = TIM_COUNTERMODE_UP
   htim1.Init.Period = COUN_MAX - 1;
   htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1
   //TIM_CLOCKDIVISION_DIV1 divides F_clock by 1
   //TIM_CLOCKDIVISION_DIV2 divides F_clock by 2
   //TIM_CLOCKDIVISION_DIV4 divides F_clock by 4
   HAL_TIM_Base_Init ( &htim1 );
   HAL_TIM_Base_Start( &htim1 );
}
```

```
// GET COUNT VALUE
uint32_t counterValue = __HAL_TIM_GET_COUNTER( &htim1 );

// COUNTER INTERRUPTION
HAL_TIM_Base_Start_IT( &htim1 );

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
   if (htim->Instance == TIMX) {
     // Action when counts up to COUT_MAX
   }
}
```

```
/////// PUERTO SERIE

//INICIALIZAR EL PUERTO SERIE
huart2.Instance = USART2;
huart2.Init.BaudRate = 9600;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
HAL_UART_Init(&huart2);


////// Transmisión y recepción

uint8_t msg[] = "Hola mundo\r\n";
HAL_UART_Transmit(&huart2, msg, sizeof(msg)-1, MAX_DELAY);

uint8_t buffer[10];
HAL_UART_Receive(&huart2, buffer, sizeof(buffer), MAX_DELAY);
```