Microcontroller Engineering TMIK13 Lecture 7

REPETITION, TIMERS, DATASHEETS DISPLAY ETC

ANDREAS AXELSSON (ANDREAS.AXELSSON@JU.SE)

Memory Map – Peripherals

Reference Manual for STM32F401 page 38-39

Table 1. STM32F401xB/C and STM32F401xD/E register boundary addresses

Boundary address	Peripheral	Bus	Register map
0x5000 0000 - 0x5003 FFFF	USB OTG FS	AHB2	*Section 22.16.6: OTG_FS register map on page 755*
0x4002 6400 - 0x4002 67FF	DMA2	AHB1	*Section 9.5.11: DMA register map on page 198*
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 3C00 - 0x4002 3FFF	Flash interface register		*Section 3.8: Flash interface registers on page 60*
0x4002 3800 - 0x4002 3BFF	RCC		*Section 6.3.22: RCC register map on page 137*
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4: CRC register map on page 70*
0x4002 1C00 - 0x4002 1FFF	GPIOH		*Section 8.4.11: GPIO register map on page 164*
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		

Defines for Peripherals

```
#define SPI1                    ((SPI_TypeDef *) SPI1_BASE)
#define SPI4                    ((SPI_TypeDef *) SPI4_BASE)
#define SYSCFG                  ((SYSCFG_TypeDef *) SYSCFG_BASE)
#define EXTI                    ((EXTI_TypeDef *) EXTI_BASE)
#define TIM9                    ((TIM_TypeDef *) TIM9_BASE)
#define TIM10                   ((TIM_TypeDef *) TIM10_BASE)
#define TIM11                   ((TIM_TypeDef *) TIM11_BASE)
#define GPIOA                   ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                   ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                   ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                   ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                   ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOH                   ((GPIO_TypeDef *) GPIOH_BASE)
#define CRC                     ((CRC_TypeDef *) CRC_BASE)
#define RCC                     ((RCC_TypeDef *) RCC_BASE)
#define FLASH                   ((FLASH_TypeDef *) FLASH_R_BASE)
#define DMA1                    ((DMA_TypeDef *) DMA1_BASE)
```

GPIO_TypeDef – stm32f401xe.h

```
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register,               Address offset: 0x00      */
  __IO uint32_t OTYPER;   /*!< GPIO port output type register,        Address offset: 0x04      */
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,       Address offset: 0x08      */
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register,  Address offset: 0x0C      */
  __IO uint32_t IDR;      /*!< GPIO port input data register,         Address offset: 0x10      */
  __IO uint32_t ODR;      /*!< GPIO port output data register,        Address offset: 0x14      */
  __IO uint32_t BSRR;     /*!< GPIO port bit set/reset register,      Address offset: 0x18      */
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C      */
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers,     Address offset: 0x20-0x24 */
} GPIO_TypeDef;
```

GPIO Register map page 164 in Reference Manual

HAL GPIO – Write Pin

```
void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
{
  /* Check the parameters */
  assert_param(IS_GPIO_PIN(GPIO_Pin));
  assert_param(IS_GPIO_PIN_ACTION(PinState));

  if(PinState != GPIO_PIN_RESET)
  {
    GPIOx->BSRR = GPIO_Pin;
  }
  else
  {
    GPIOx->BSRR = (uint32_t)GPIO_Pin << 16U;
  }
}
```

HAL GPIO – BSRR

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..E and H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

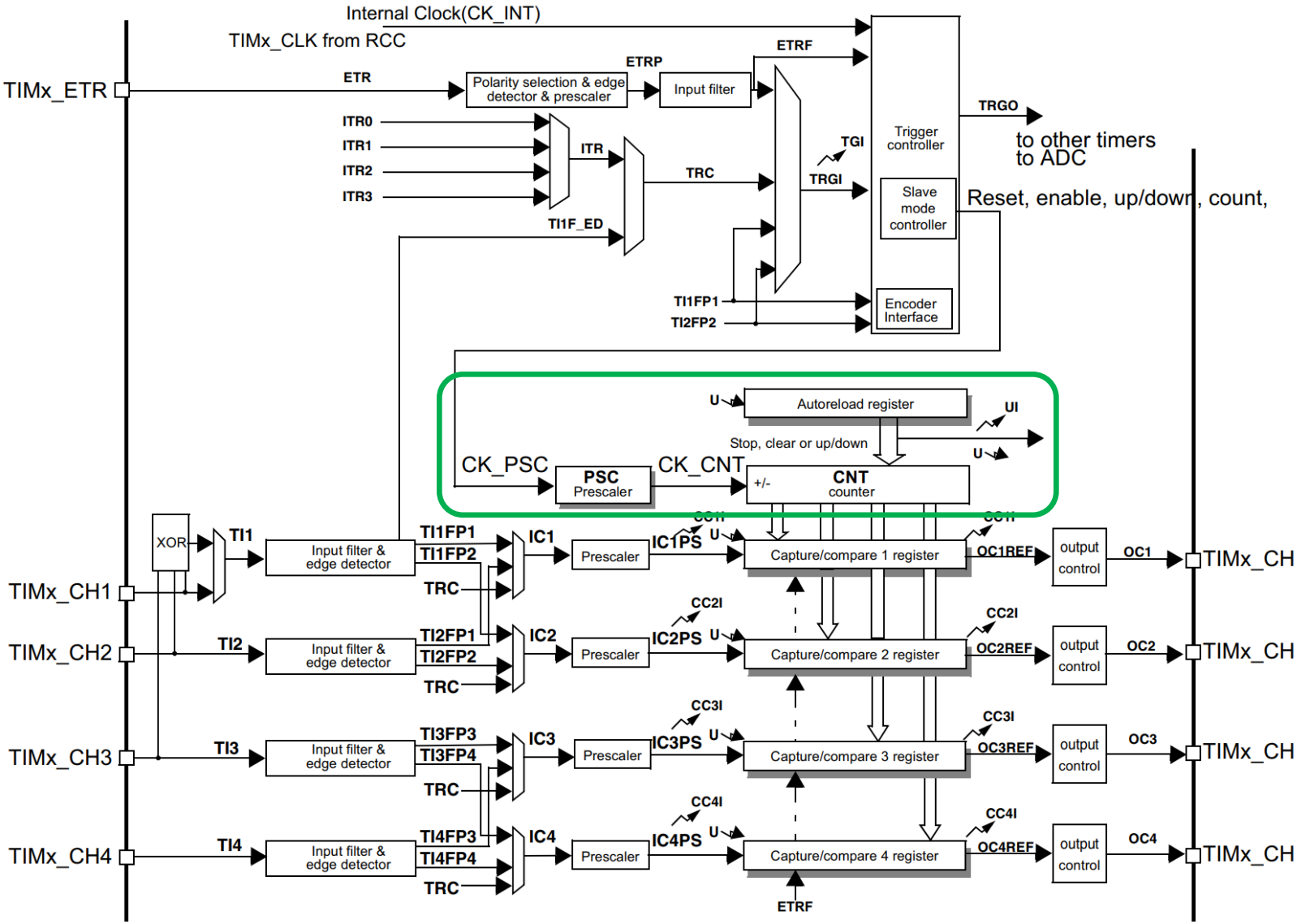
Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

Timer – Block Schematics



Memory Map – Peripherals

Table 1. STM32F401xB/C and STM32F401xD/E register boundary addresses (continued)

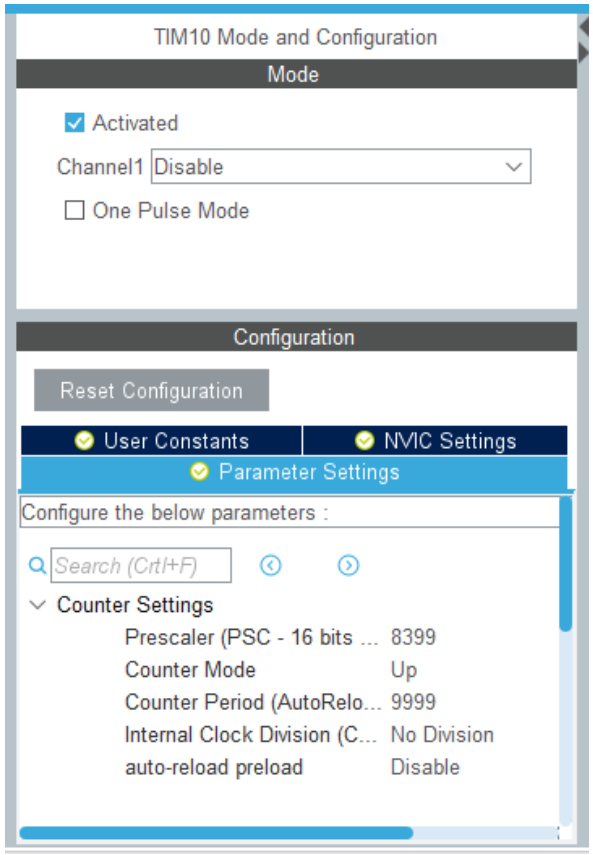
Boundary address	Peripheral	Bus	Register map
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	*Section 14.5.12: TIM10/11 register map on page 420*
0x4001 4400 - 0x4001 47FF	TIM10		
0x4001 4000 - 0x4001 43FF	TIM9		*Section 14.4.13: TIM9 register map on page 410*
0x4001 3C00 - 0x4001 3FFF	EXTI		*Section 10.3.7: EXTI register map on page 212*
0x4001 3800 - 0x4001 3BFF	SYSCFG		*Section 7.2.8: SYSCFG register map*
0x4001 3400 - 0x4001 37FF	SPI4		*Section 20.5.10: SPI register map on page 611*
0x4001 3000 - 0x4001 33FF	SPI1		
0x4001 2C00 - 0x4001 2FFF	SDIO		*Section 21.9.16: SDIO register map on page 667*
0x4001 2000 - 0x4001 23FF	ADC1		*Section 11.12.16: ADC register map on page 240*
0x4001 1400 - 0x4001 17FF	USART6		*Section 19.6.8: USART register map on page 558*
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0000 - 0x4001 03FF	TIM1		*Section 12.4.21: TIM1 register map on page 314*
0x4000 7000 - 0x4000 73FF	PWR	APB1	*Section 5.5: PWR register map on page 90*
0x4000 5C00 - 0x4000 5FFF	I2C3		*Section 18.6.11: I2C register map on page 505*
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 4400 - 0x4000 47FF	USART2		*Section 19.6.8: USART register map on page 558*
0x4000 4000 - 0x4000 43FF	I2S3ext		*Section 20.5.10: SPI register map on page 611*
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		*Section 15.4.5: IWDG register map on page 426*
0x4000 2C00 - 0x4000 2FFF	WWDG		*Section 16.6.4: WWDG register map on page 433*
0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers		*Section 17.6.21: RTC register map on page 471*
0x4000 0C00 - 0x4000 0FFF	TIM5		*Section 13.4.21: TIMx register map on page 374*
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		

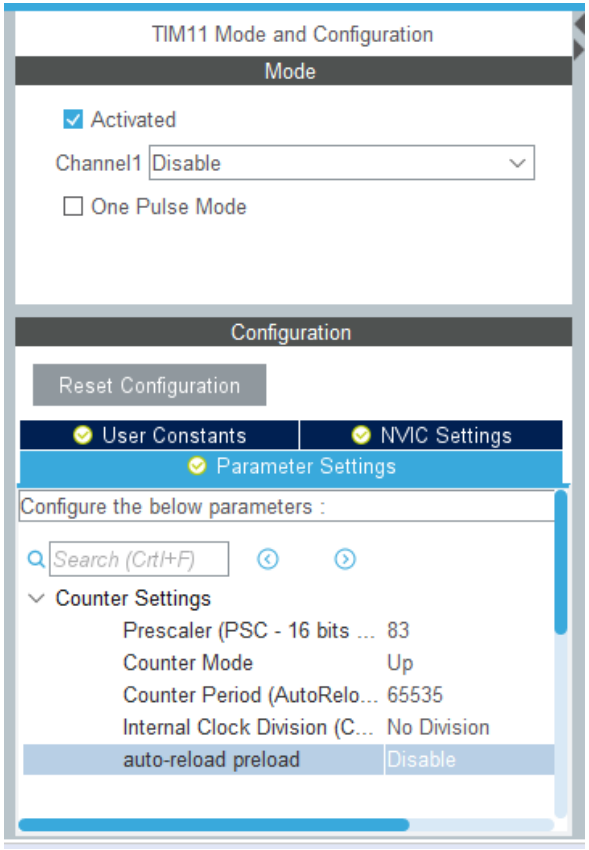
TIM_TypeDef – stm32f401xe.h

```
typedef struct
{
  __IO uint32_t CR1;         /*!< TIM control register 1,              Address offset: 0x00 */
  __IO uint32_t CR2;         /*!< TIM control register 2,              Address offset: 0x04 */
  __IO uint32_t SMCR;        /*!< TIM slave mode control register,     Address offset: 0x08 */
  __IO uint32_t DIER;        /*!< TIM DMA/interrupt enable register,   Address offset: 0x0C */
  __IO uint32_t SR;          /*!< TIM status register,                 Address offset: 0x10 */
  __IO uint32_t EGR;         /*!< TIM event generation register,       Address offset: 0x14 */
  __IO uint32_t CCMR1;       /*!< TIM capture/compare mode register 1, Address offset: 0x18 */
  __IO uint32_t CCMR2;       /*!< TIM capture/compare mode register 2, Address offset: 0x1C */
  __IO uint32_t CCER;        /*!< TIM capture/compare enable register, Address offset: 0x20 */
  __IO uint32_t CNT;         /*!< TIM counter register,                Address offset: 0x24 */
  __IO uint32_t PSC;         /*!< TIM prescaler,                       Address offset: 0x28 */
  __IO uint32_t ARR;         /*!< TIM auto-reload register,            Address offset: 0x2C */
  __IO uint32_t RCR;         /*!< TIM repetition counter register,     Address offset: 0x30 */
  __IO uint32_t CCR1;        /*!< TIM capture/compare register 1,      Address offset: 0x34 */
  __IO uint32_t CCR2;        /*!< TIM capture/compare register 2,      Address offset: 0x38 */
  __IO uint32_t CCR3;        /*!< TIM capture/compare register 3,      Address offset: 0x3C */
  __IO uint32_t CCR4;        /*!< TIM capture/compare register 4,      Address offset: 0x40 */
  __IO uint32_t BDTR;        /*!< TIM break and dead-time register,    Address offset: 0x44 */
  __IO uint32_t DCR;         /*!< TIM DMA control register,            Address offset: 0x48 */
  __IO uint32_t DMAR;        /*!< TIM DMA address for full transfer,   Address offset: 0x4C */
  __IO uint32_t OR;          /*!< TIM option register,                 Address offset: 0x50 */
} TIM_TypeDef;
```

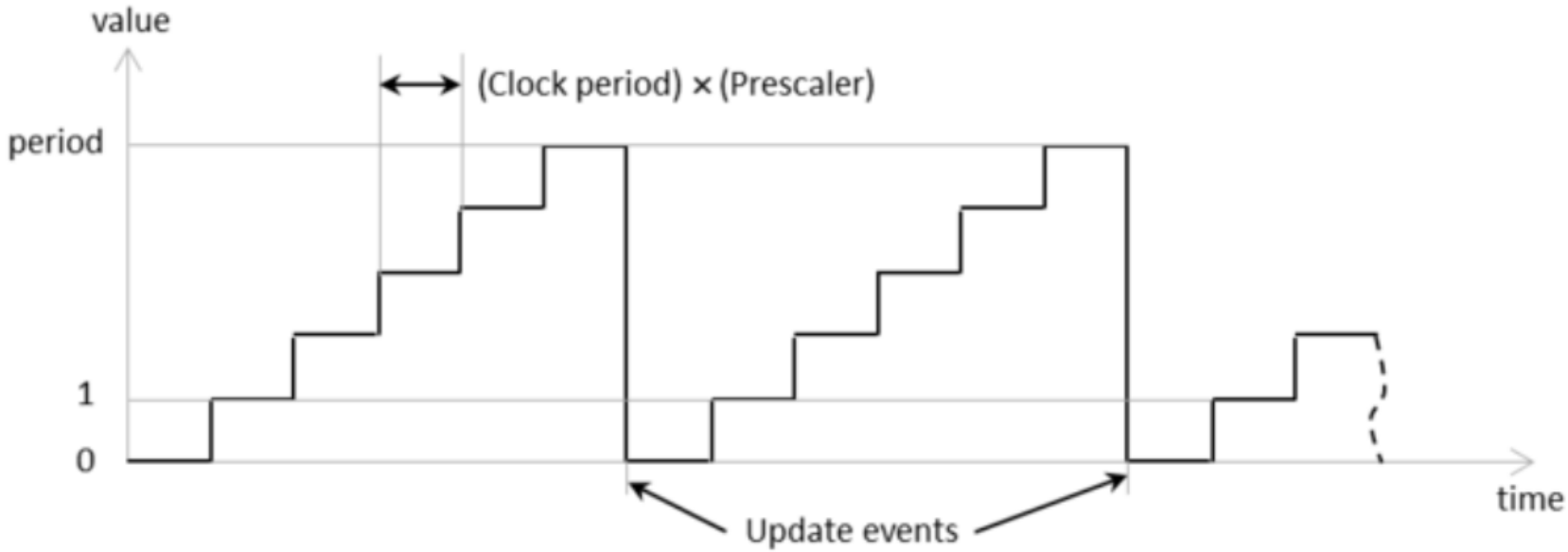
TIM Register maps found in Reference Manual

Timer basic setup





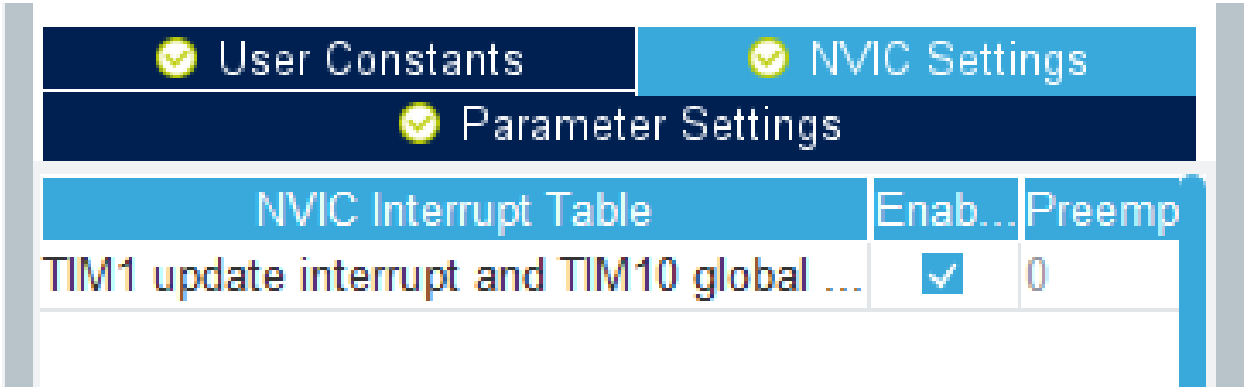
Timer period and elapsed time




```
htim10.Instance = TIM10;
htim10.Init.Prescaler = 8399;
htim10.Init.CounterMode = TIM_COUNTERMODE_UP;
htim10.Init.Period = 9999;
htim10.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim10.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim10) != HAL_OK)
{
  Error_Handler();
}
```

https://visualgdb.com/tutorials/arm/stm32/timers/hal/

Timer interrupts / callbacks



stm32f4xx_it.c

stm32f4xx_hal_tim.c

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM10)
    {
        // Do things
    }
    if (htim->Instance == TIM11)
    {
        // Do other things
    }

}
```

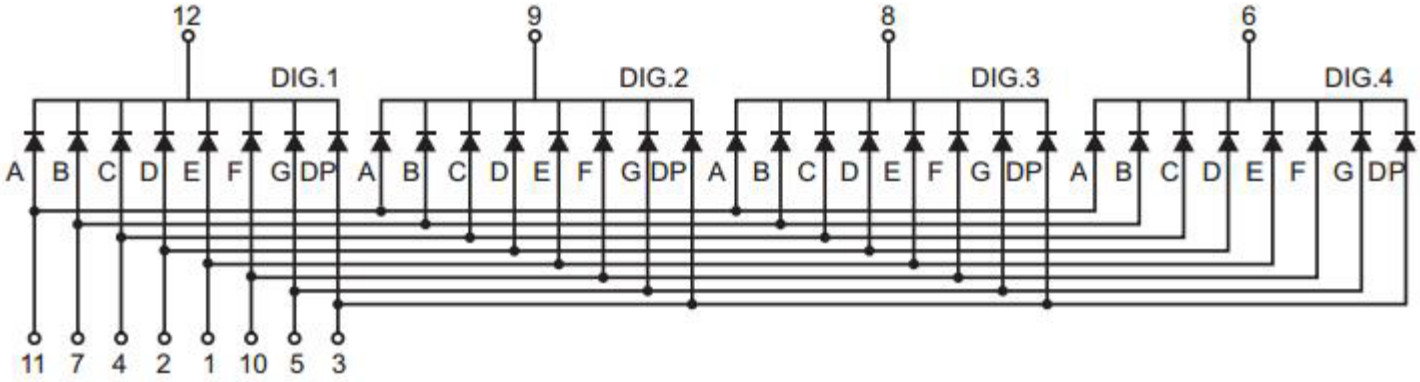
Timer – Read counter value

```
__IO uint32_t SR;          /*!< TIM status register,                  Address offset: 0x10 */
__IO uint32_t EGR;         /*!< TIM event generation register,        Address offset: 0x14 */
__IO uint32_t CCMR1;       /*!< TIM capture/compare mode register 1,  Address offset: 0x18 */
__IO uint32_t CCMR2;       /*!< TIM capture/compare mode register 2,  Address offset: 0x1C */
__IO uint32_t CCER;        /*!< TIM capture/compare enable register,  Address offset: 0x20 */
__IO uint32_t CNT;         /*!< TIM counter register,                 Address offset: 0x24 */
__IO uint32_t PSC;         /*!< TIM prescaler,                        Address offset: 0x28 */
__IO uint32_t ARR;         /*!< TIM auto-reload register,             Address offset: 0x2C */
__IO uint32_t RCR;         /*!< TIM repetition counter register,      Address offset: 0x30 */
__IO uint32_t CCR1;        /*!< TIM capture/compare register 1,       Address offset: 0x34 */
__IO uint32_t CCR2;        /*!< TIM capture/compare register 2,       Address offset: 0x38 */
```

```
uint32_t count = __HAL_TIM_GET_COUNTER(&htim10);
```

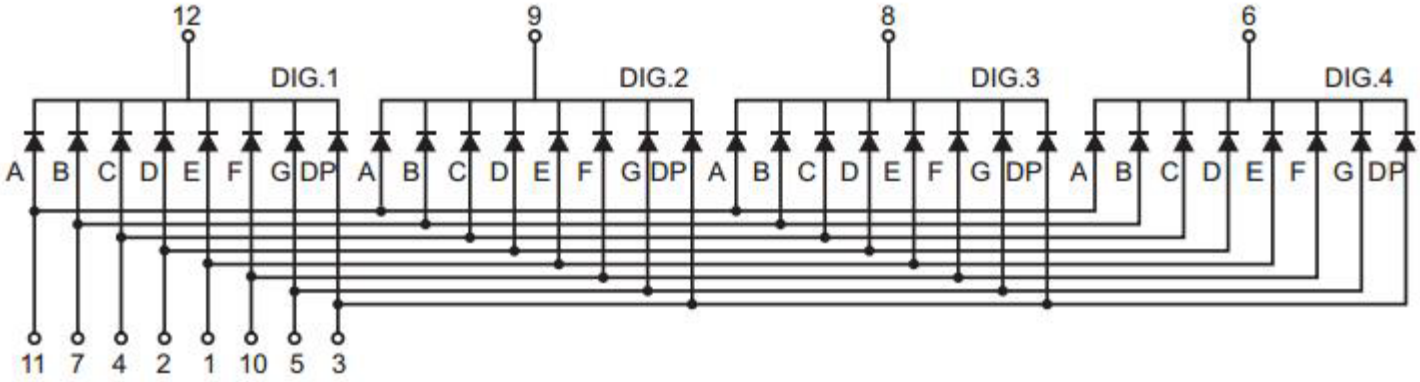
```
/**
  * @brief  Get the TIM Counter Register value on runtime.
  * @param  __HANDLE__ TIM handle.
  * @retval 16-bit or 32-bit value of the timer counter register (TIMx_CNT)
  */
#define __HAL_TIM_GET_COUNTER(__HANDLE__)  ((__HANDLE__)->Instance->CNT)
```

4-digit 7-Segment



4-digit 7-Segment

Your devices has a driver IC attached to your 7-seg. A TM1637. It uses a serial protocol to set the display numbers. See **quad_sseg.zip** for driver code.



16 x 2 LCD Display

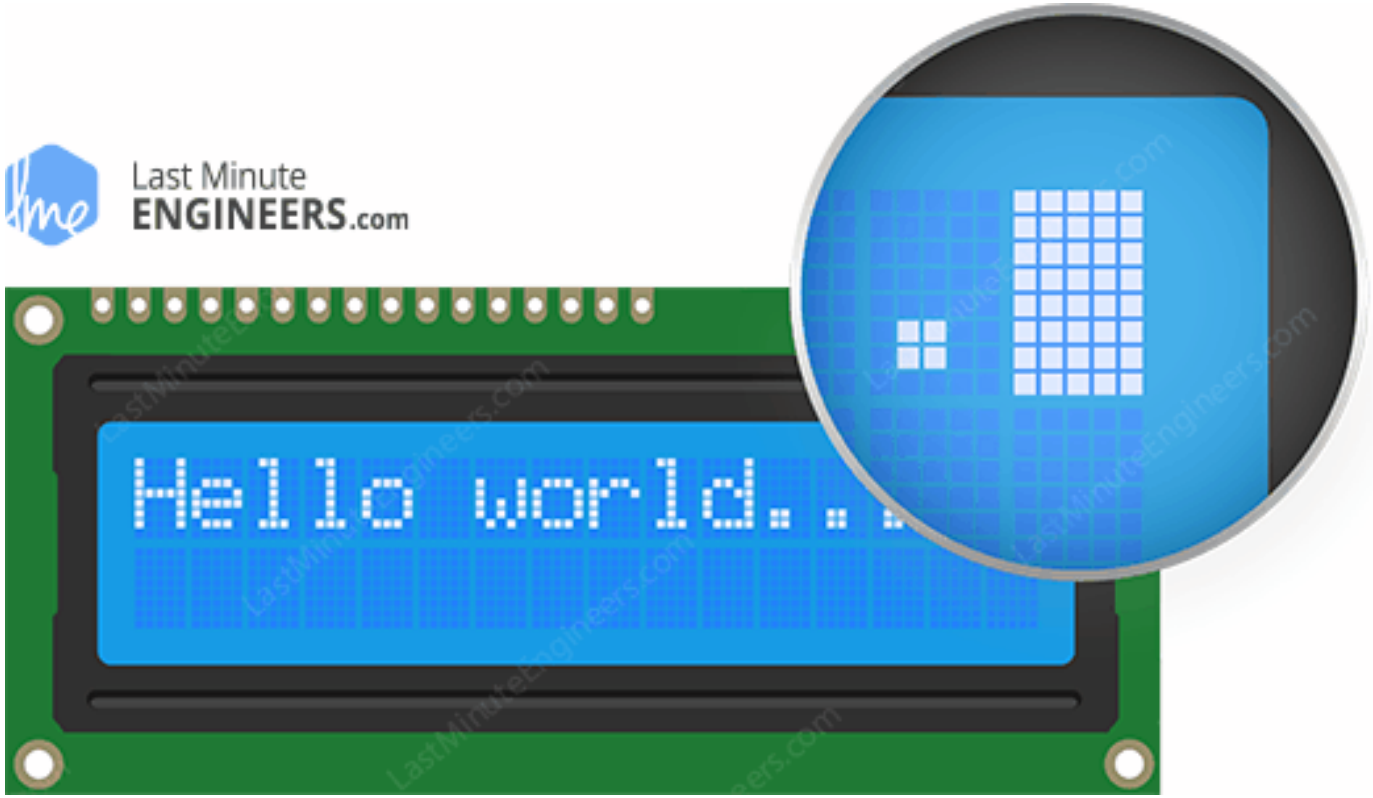
LCD display with 16 x 2 characters

Uses a HD44780 compatible controller

Backlight (A-K connectors)

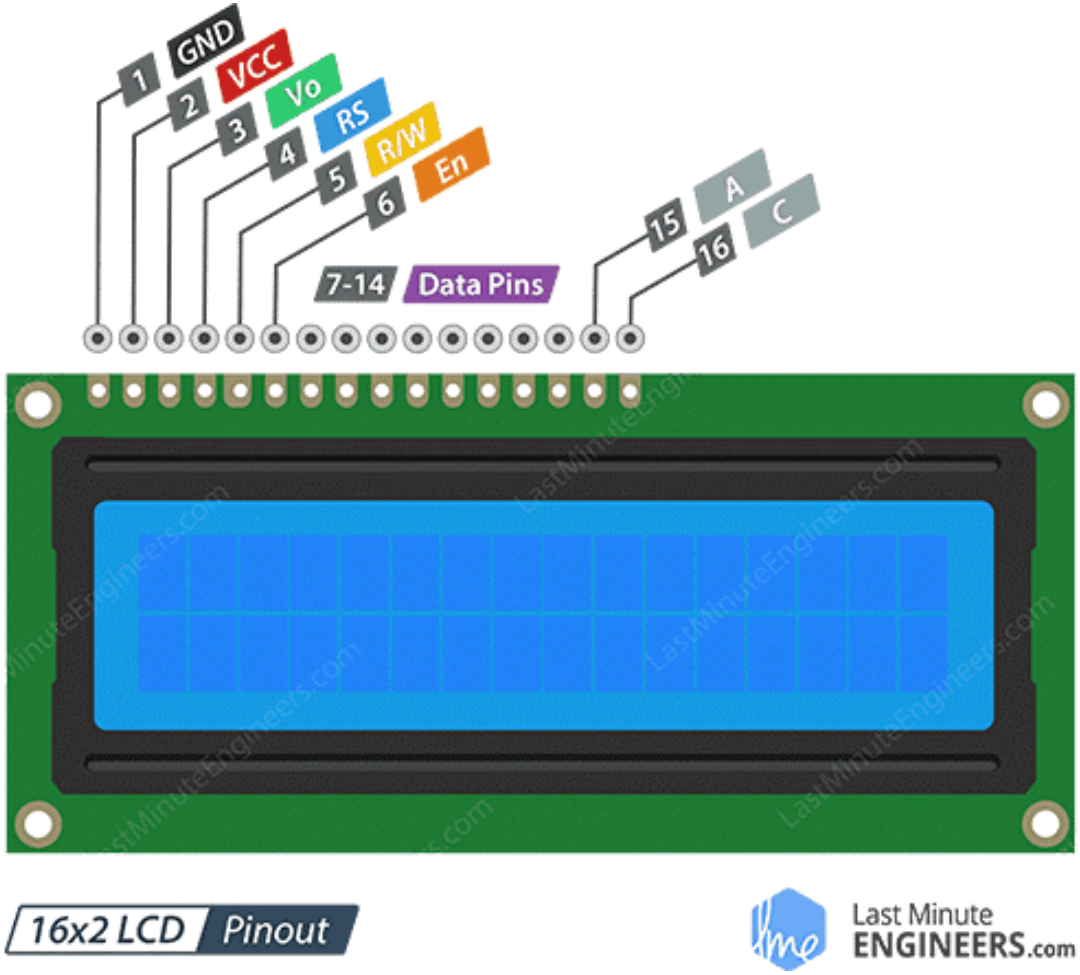
RS/RW/E control signals and D0-D7 data pins (4 or 8 bit mode)

16 x 2 LCD Display



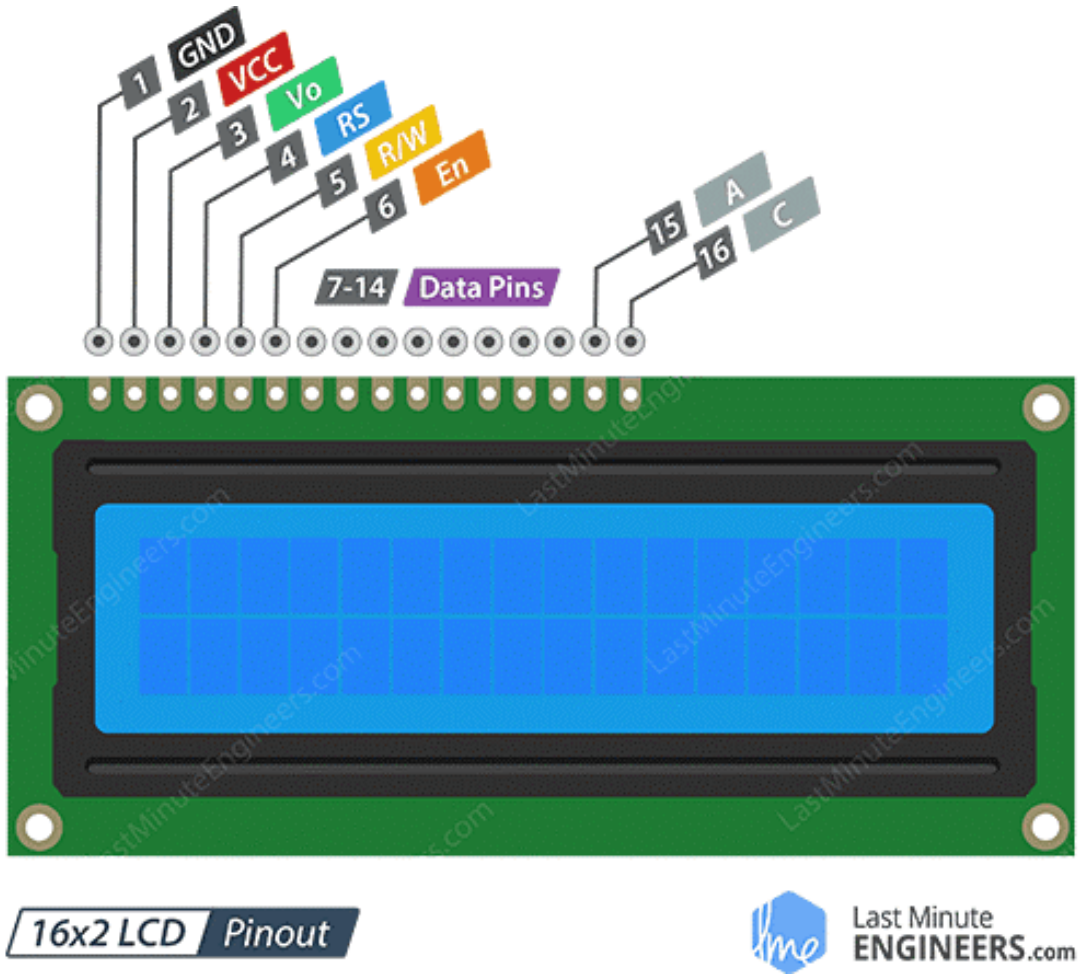
https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/

16 x 2 LCD Display



https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/

16 x 2 LCD Display



The LCD display in your kit has a special interface driver circuit using I2C to communicate with the LCD interpace.

It can write 8 bits via I2C to the RS R/W EN and D3-D0 pins on the LCD display.
Thus the display is then operated in 4-bit mode.
Please look in the datasheet to see the difference between 8-bit and 4-bit mode.

https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/

HD44780U

HD44780U based instruction set

Instruction	Code										Description	Execution time (max) (when f_{cp} = 270 kHz)
	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.52 ms
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position. Also returns display being shifted to the original position. DDRAM content remains unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D); specifies to shift the display (S). These operations are performed during data read/write.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets on/off of all display (D), cursor on/off (C), and blink of cursor position character (B).	37 µs
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM content remains unchanged.	37 µs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data are sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data are sent and received after this setting.	37 µs
Read busy flag & address counter	0	1	BF	CGRAM/DDRAM address							Reads busy flag (BF) indicating internal operation being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0 µs
Write CGRAM or DDRAM	1	0	Write Data								Write data to CGRAM or DDRAM.	37 µs
Read from CG/DDRAM	1	1	Read Data								Read data from CGRAM or DDRAM.	37 µs

Instruction bit names —

I/D - 0 = decrement cursor position, 1 = increment cursor position; **S** - 0 = no display shift, 1 = display shift; **D** - 0 = display off, 1 = display on; **C** - 0 = cursor off, 1 = cursor on; **B** - 0 = cursor blink off, 1 = cursor blink on ; **S/C** - 0 = move cursor, 1 = shift display; **R/L** - 0 = shift left, 1 = shift right; **DL** - 0 = 4-bit interface, 1 = 8-bit interface; **N** - 0 = 1/8 or 1/11 duty (1 line), 1 = 1/16 duty (2 lines); **F** - 0 = 5×8 dots, 1 = 5×10 dots; **BF** - 0 = can accept instruction, 1 = internal operation in progress.

https://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller

8-bit Initialization

8 - Bit Initialization:

Power ON

Wait more than 15ms after Vcc = 4.5V

No data should be transferred to or from the display during this time.

RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
0	0	0	0	1	1	x	x	x	x

Function Set Command: (8-Bit interface)
BF cannot be checked before this command.

Wait more than 4.1ms

No data should be transferred to or from the display during this time.

RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
0	0	0	0	1	1	x	x	x	x

Function Set Command: (8-Bit interface)
BF cannot be checked before this command.

Wait more than 100÷s

No data should be transferred to or from the display during this time.

RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
0	0	0	0	1	1	x	x	x	x

Function Set Command: (8-Bit interface)
After this command is written, BF can be checked.

RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	
0	0	0	0	1	1	N	F	x	x	Function Set (Interface = 8 bits, Set No. of lines and display font)
0	0	0	0	0	0	1	0	0	0	Display OFF
0	0	0	0	0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	0	1	I/D	S	Entry Mode Set:
0	0	0	0	0	0	1	1	C	B	Display ON (Set C and B for cursor/Blink options.)

Initialization Complete,
Display Ready.

Note: BF should be checked before each of the instructions starting with Display OFF.

.c and .h files

.h – header file

- Defines the function prototypes
- Included in files that uses the specified functions
- #include ”myheaderfile.h” at top of .c file

.c – Implementation file

- Contains the code of the functions
- Functions without prototype in .h file is considered ”private”

LCD Driver (lcd.c, lcd.h)

- `void TextLCD_Init(TextLCDType *lcd, GPIO_TypeDef *controlPort,`

  `uint16_t rsPin, uint16_t rwPin, uint16_t enPin,`

  `GPIO_TypeDef *dataPort, uint16_t dataPins)`
- `void TextLCD_Home(TextLCDType *lcd)`
- `void TextLCD_Clear(TextLCDType *lcd)`
- `void TextLCD_Position(TextLCDType *lcd, int x, int y)`
- `void TextLCD_Putchar(TextLCDType *lcd, uint8_t data)`
- `void TextLCD_Puts(TextLCDType *lcd, char *string)`
- `void TextLCD_Printf(TextLCDType *lcd, char *message, …)`

LCD Driver (lcd.c)

- `void TextLCD_Strobe(TextLCDType *lcd)`
- `void TextLCD_Cmd(TextLCDType *lcd, uint8_t cmd)`
- `void TextLCD_Data(TextLCDType *lcd, uint8_t data)`

What is the purpose of the functions above?

What about TextLCDType?

```
typedef struct {
…
} TextLCDType;
```

ASCII Table

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Microcontroller Engineering

Questions?

Contact information

Andreas Axelsson

Email: andreas.axelsson@ju.se

Mobile: 0709-467760