PERIPHERALS

EL-GY 6483 Real Time Embedded Systems

PERIPHERALS

- Buttons and keypads
- Timers
- Sensors (temperature, light, microphone, pressure)
- Actuators (motors, e.g. servo)
- Displays (individual LEDs, seven segment displays, pixel displays)
- Communication (SPI, I2C, UART, USB, Ethernet/WiFi/Bluetooth/Zigbee)

USING I/O

- Memory-mapped I/O: the memory and registers of the I/O devices are mapped to address values (uses same address bus), same instructions used for accessing memory are used for accessing I/O devices.
- Port-mapped (isolated) I/O: write to the registers of the I/O interface through the data (not address) bus, which later sends the data to the actual I/O device. Often has special instructions for accessing I/O devices.

USING I/O

Memory-mapped IO	Port-mapped IO
Same address bus to address memory and I/O devices	Different address spaces for memory and I/O devices
Access to the I/O devices using regular instructions	Uses a special class of CPU instructions to access I/O devices
Most widely used I/O method	x86 Intel microprocessors - IN and OUT instructions

EXAMPLE: BLINKY

On MK20DX256VLH7 (Cortex M4):

- Program memory (Flash) begins at 0x00000000
- SRAM (stack) begins at 1FFC0000 (0x20000000 – 0x40000000)
- Peripherals begin at 0x40000000
- GPIOD port uses 0x400FF000 - 0x400FF114

EXAMPLE: BLINKY

r0	0x400FF094	1074786452
r1	0x00000020	32
r2	0x00300000	3145728
r3	0x0	0
r4	0x0	0
r5	0x0	0
r6	0x0	0
r7	0x0	0
r8	0x0	0
r9	0x0	0
r10	0x0	0
r11	0x0	0
r12	0x0	0
sp	0x1fff803c	536838204
lr	0x00003fd	1021
pc	0x000040f	1039

GPIO

GPIO

GPIO pin: Generic pin whose behavior, including whether it is an input or output pin, can be controlled by the user at run time.

A GPIO port is a group of GPIO pins.

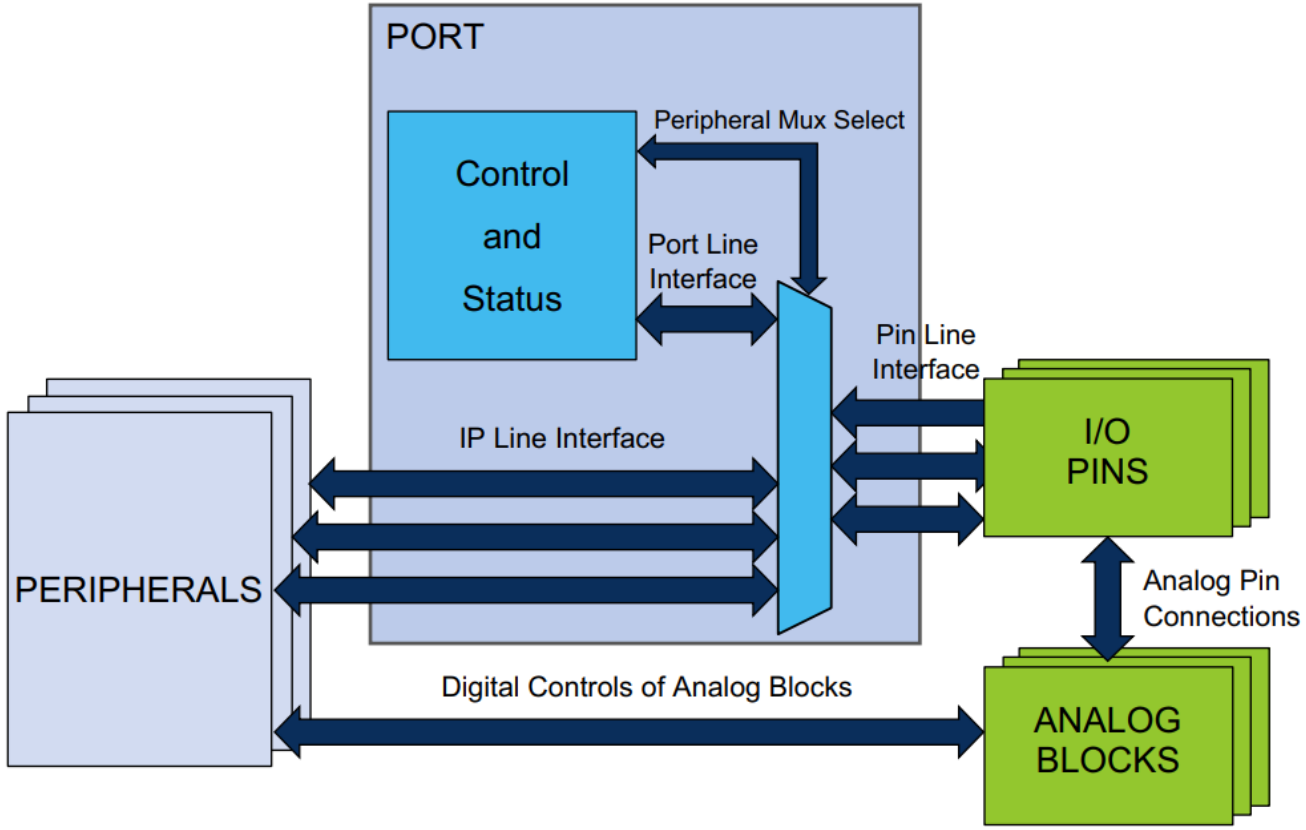
HARDWARE ABSTRACTION LAYER (HAL)

A hardware abstraction layer is a programming API to interact with peripherals.

- CMSIS (Cortex Microcontroller Software Interface Standard) is a vendor independent HAL for using peripherals on the Cortex-M processor series.
- Environments may provide several HALs for using peripherals on Arm Cortex series microcontrollers.

GPIO on M0+ (ATSAMD21)

- PORT Block Diagram



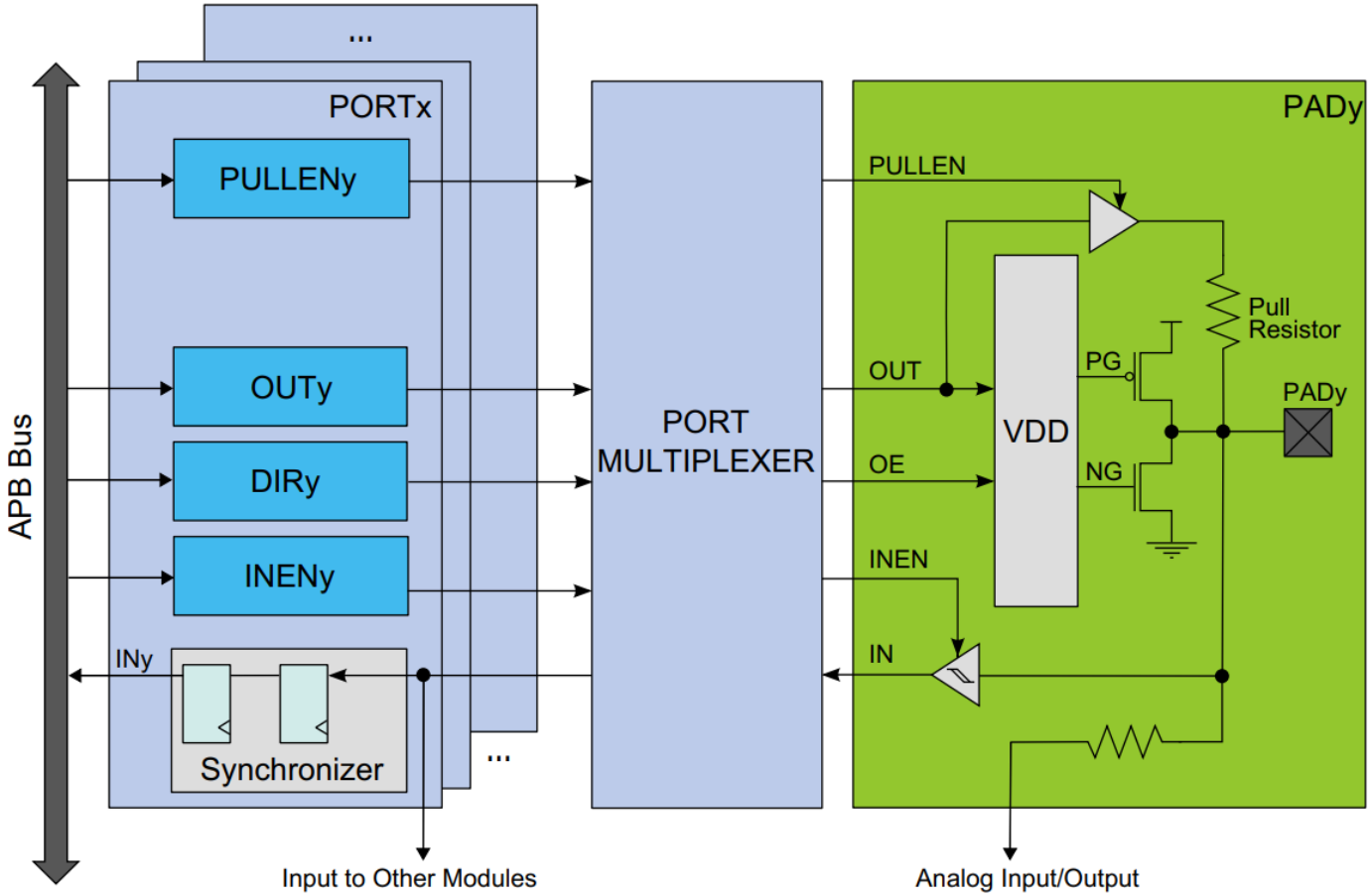
Alternative Functions on M0+ (ATSAMD21)

- Pin Muxing Cross Referrence

Pin[1]			I/O Pin	Supply	Type	A	B[2][3]					C	D	E	F	G	H
SAMD21E	SAMD21G	SAMD21J				EIC	REF	ADC	AC	PTC	DAC	SERCOM[2][3]	SERCOM-ALT	TC[4]/TCC	TCC	COM	AC/GCLK
	19	23	PB10	VDDIO		EXTINT[10]							SERCOM4/PAD[2]	TC5/WO[0]	TCC0/WO[4]	I2S/MCK[1]	GCLK_IO[4]
	20	24	PB11	VDDIO		EXTINT[11]							SERCOM4/PAD[3]	TC5/WO[1]	TCC0/WO[5]	I2S/SCK[1]	GCLK_IO[5]
		25	PB12	VDDIO	I²C	EXTINT[12]				X[12]		SERCOM4/PAD[0]		TC4/WO[0]	TCC0/WO[6]	I2S/FS[1]	GCLK_IO[6]
		26	PB13	VDDIO	I²C	EXTINT[13]				X[13]		SERCOM4/PAD[1]		TC4/WO[1]	TCC0/WO[7]		GCLK_IO[7]
		27	PB14	VDDIO		EXTINT[14]				X[14]		SERCOM4/PAD[2]		TC5/WO[0]			GCLK_IO[0]
		28	PB15	VDDIO		EXTINT[15]				X[15]		SERCOM4/PAD[3]		TC5/WO[1]			GCLK_IO[1]
	21	29	PA12	VDDIO	I²C	EXTINT[12]						SERCOM2/PAD[0]	SERCOM4/PAD[0]	TCC2/WO[0]	TCC0/WO[6]		AC/CMP[0]
	22	30	PA13	VDDIO	I²C	EXTINT[13]						SERCOM2/PAD[1]	SERCOM4/PAD[1]	TCC2/WO[1]	TCC0/WO[7]		AC/CMP[1]
15	23	31	PA14	VDDIO		EXTINT[14]						SERCOM2/PAD[2]	SERCOM4/PAD[2]	TC3/WO[0]	TCC0/WO[4]		GCLK_IO[0]
16	24	32	PA15	VDDIO		EXTINT[15]						SERCOM2/PAD[3]	SERCOM4/PAD[3]	TC3/WO[1]	TCC0/WO[5]		GCLK_IO[1]
17	25	35	PA16	VDDIO	I²C	EXTINT[0]				X[4]		SERCOM1/PAD[0]	SERCOM3/PAD[0]	TCC2/WO[0]	TCC0/WO[6]		GCLK_IO[2]
18	26	36	PA17	VDDIO	I²C	EXTINT[1]				X[5]		SERCOM1/PAD[1]	SERCOM3/PAD[1]	TCC2/WO[1]	TCC0/WO[7]		GCLK_IO[3]
19	27	37	PA18	VDDIO		EXTINT[2]				X[6]		SERCOM1/PAD[2]	SERCOM3/PAD[2]	TC3/WO[0]	TCC0/WO[2]		AC/CMP[0]
20	28	38	PA19	VDDIO		EXTINT[3]				X[7]		SERCOM1/PAD[3]	SERCOM3/PAD[3]	TC3/WO[1]	TCC0/WO[3]	I2S/SD[0]	AC/CMP[1]
		39	PB16	VDDIO	I²C	EXTINT[0]						SERCOM5/PAD[0]		TC6/WO[0]	TCC0/WO[4]	I2S/SD[1]	GCLK_IO[2]
		40	PB17	VDDIO	I²C	EXTINT[1]						SERCOM5/PAD[1]		TC6/WO[1]	TCC0/WO[5]	I2S/MCK[0]	GCLK_IO[3]

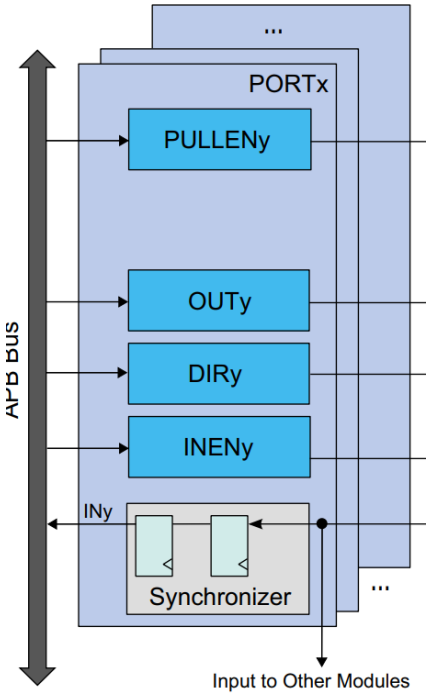
PORT Function on M0+ (ATSAMD21)

- Pin Muxing Cross Referrence

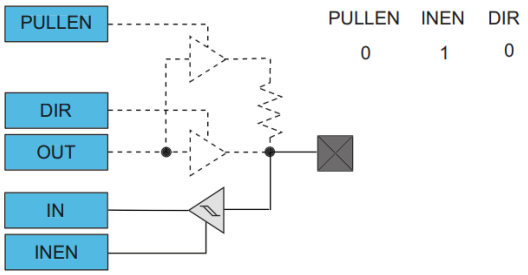


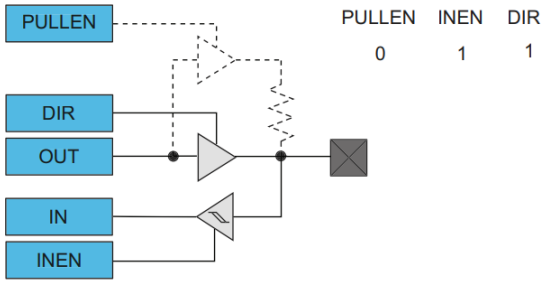
Pin Configuration

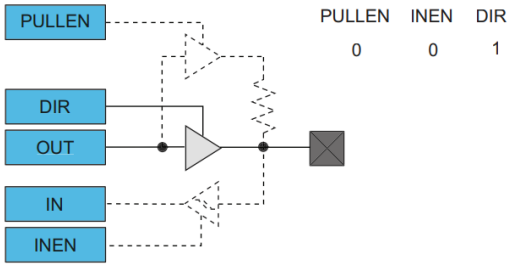
DIR	INEN	PULLEN	OUT	Configuration
0	0	0	X	Reset or analog I/O; all digital disabled
0	0	1	0	Pull-down; input disabled
0	0	1	1	Pull-up; input disabled
0	1	0	X	Input
0	1	1	0	Input with pull-down
0	1	1	1	Input with pull-up
1	0	X	X	Output; input disabled
1	1	X	X	Output; input enabled

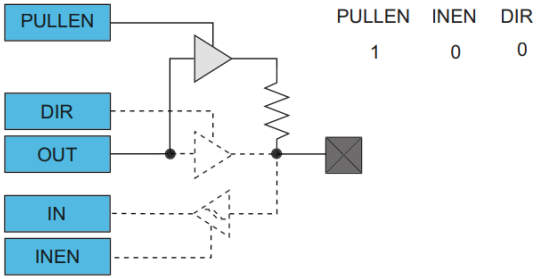


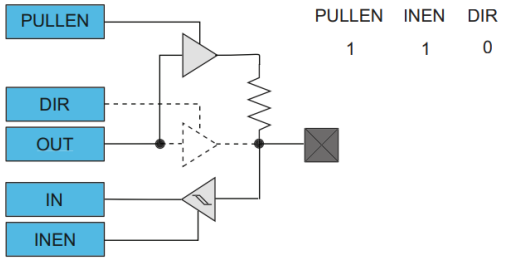
Pin Configuration

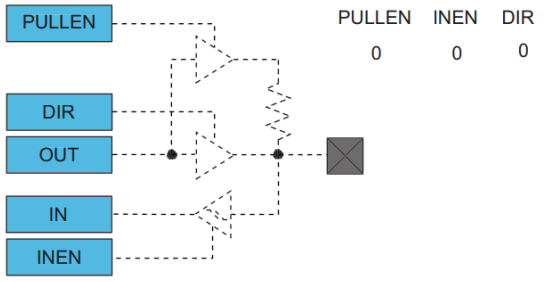












Pin Control REGISTERS - DIR

Offset	Name	Bit Pos.								
0x00	DIR	7:0	DIR[7:0]							
0x01		15:8	DIR[15:8]							
0x02		23:16	DIR[23:16]							
0x03		31:24	DIR[31:24]							
0x04	DIRCLR	7:0	DIRCLR[7:0]							
0x05		15:8	DIRCLR[15:8]							
0x06		23:16	DIRCLR[23:16]							
0x07		31:24	DIRCLR[31:24]							
0x08	DIRSET	7:0	DIRSET[7:0]							
0x09		15:8	DIRSET[15:8]							
0x0A		23:16	DIRSET[23:16]							
0x0B		31:24	DIRSET[31:24]							

Pin Control REGISTERS - DIR

Data Direction

Name: DIR
Offset: 0x00+x*0x80 [x=0..2]
Reset: 0x00000000
Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	DIR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 31:0 – DIR[31:0]: Port Data Direction**
  These bits set the data direction for the individual I/O pins in the PORT group.
  0: The corresponding I/O pin in the group is configured as an input.
  1: The corresponding I/O pin in the group is configured as an output.

Pin Control REGISTERS

```
/* ========== Register definition for PORT peripheral ========== */
#if (defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_ASM__))
#define REG_PORT_DIR0              (0x41004400U) /**< \brief (PORT) Data Direction 0 */
#define REG_PORT_DIRCLR0           (0x41004404U) /**< \brief (PORT) Data Direction Clear 0 */
#define REG_PORT_DIRSET0           (0x41004408U) /**< \brief (PORT) Data Direction Set 0 */
#define REG_PORT_DIRTGL0           (0x4100440CU) /**< \brief (PORT) Data Direction Toggle 0 */
#define REG_PORT_OUT0              (0x41004410U) /**< \brief (PORT) Data Output Value 0 */
#define REG_PORT_OUTCLR0           (0x41004414U) /**< \brief (PORT) Data Output Value Clear 0 */
#define REG_PORT_OUTSET0           (0x41004418U) /**< \brief (PORT) Data Output Value Set 0 */
#define REG_PORT_OUTTGL0           (0x4100441CU) /**< \brief (PORT) Data Output Value Toggle 0 */
#define REG_PORT_IN0               (0x41004420U) /**< \brief (PORT) Data Input Value 0 */
#define REG_PORT_CTRL0             (0x41004424U) /**< \brief (PORT) Control 0 */
#define REG_PORT_WRCONFIG0         (0x41004428U) /**< \brief (PORT) Write Configuration 0 */
#define REG_PORT_PMUX0             (0x41004430U) /**< \brief (PORT) Peripheral Multiplexing 0 */
#define REG_PORT_PINCFG0           (0x41004440U) /**< \brief (PORT) Pin Configuration 0 */
#define REG_PORT_DIR1              (0x41004480U) /**< \brief (PORT) Data Direction 1 */
#define REG_PORT_DIRCLR1           (0x41004484U) /**< \brief (PORT) Data Direction Clear 1 */
#define REG_PORT_DIRSET1           (0x41004488U) /**< \brief (PORT) Data Direction Set 1 */
#define REG_PORT_DIRTGL1           (0x4100448CU) /**< \brief (PORT) Data Direction Toggle 1 */
```

// set pin 5 on port 0 to Output mode

*REG_PORT_DIR0 |= 0x00000100; //See HAL

Pin Control REGISTERS

Name: WRCONFIG
Offset: 0x28+x*0x80 [x=0..2]
Reset: 0x00000000
Property: Write-Protected

Bit	31	30	29	28	27	26	25	24
	HWSEL	WRPINCFG		WRPMUX	PMUX[3:0]			
Access	W	W	R	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		DRVSTR				PULLEN	INEN	PMUXEN
Access	R	W	R	R	R	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PINMASK[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PINMASK[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

- **Bit 31 – HWSEL: Half-Word Select**
  This bit selects the half-word field of a 32-pin group to be reconfigured in the atomic write operation.
  0: The lower 16 pins of the PORT group will be configured.
  1: The upper 16 pins of the PORT group will be configured.
  This bit will always read as zero.
- **Bit 30 – WRPINCFG: Write PINCFG**
  This bit determines whether the atomic write operation will update the Pin Configuration register (PINCFGy) or not for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits.
  0: The PINCFGy registers of the selected pins will not be updated.
  1: The PINCFGy registers of the selected pins will be updated.
  Writing a zero to this bit has no effect.

Writing a one to this bit updates the configuration of the selected pins with the written WRCONFIG.DRVSTR, WRCONFIG.SLEWLIM, WRCONFIG.ODRAIN, WRCONFIG.PULLEN, WRCONFIG.INEN, WRCONFIG.PMUXEN and WRCONFIG.PINMASK values.
This bit will always read as zero.

Bit 29 – Reserved
This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

Bit 28 – WRPMUX: Write PMUX
This bit determines whether the atomic write operation will update the Peripheral Multiplexing register (PMUXn) or not for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits.
0: The PMUXn registers of the selected pins will not be updated.
1: The PMUXn registers of the selected pins will be updated.
Writing a zero to this bit has no effect.
Writing a one to this bit updates the pin multiplexer configuration of the selected pins with the written WRCONFIG.PMUX value.
This bit will always read as zero.

Bits 27:24 – PMUX[3:0]: Peripheral Multiplexing
These bits determine the new value written to the Peripheral Multiplexing register (PMUXn) for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits, when the WRCONFIG.WRPMUX bit is set.
These bits will always read as zero.

Bit 23 – Reserved
This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

Bit 22 – DRVSTR: Output Driver Strength Selection
This bit determines the new value written to PINCFGy.DRVSTR for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits the WRCONFIG.WRPINCFG bit is set.
This bit will always read as zero.

Bits 21:19 – Reserved
These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.

Bit 18 – PULLEN: Pull Enable
This bit determines the new value written to PINCFGy.PULLEN for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits when the WRCONFIG.WRPINCFG bit is set.
This bit will always read as zero.

Bit 17 – INEN: Input Enable
This bit determines the new value written to PINCFGy.DRVSTR for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits when the WRCONFIG.WRPINCFG bit is set.
This bit will always read as zero.

Bit 16 – PMUXEN: Peripheral Multiplexer Enable
This bit determines the new value written to PINCFGy.PMUXEN for all pins selected by the WRCONFIG.PINMASK and WRCONFIG.HWSEL bits when the WRCONFIG.WRPINCFG bit is set.
This bit will always read as zero.

Bits 15:0 – PINMASK[15:0]: Pin Mask for Multiple Pin Configuration
These bits select the pins to be configured within the half-word group selected by the WRCONFIG.HWSEL bit.
0: The configuration of the corresponding I/O pin in the half-word group will be left unchanged.

ADDITIONAL PIN CONFIGURATION

Peripheral Multiplexing n

There are up to 16 Peripheral Multiplexing registers in each group, one for every set of two subsequent I/O lines. The n denotes the number of the set of I/O lines, while the x denotes the number of the group.

Name: PMUXn

Offset: 0x30+n [n=0..15]+x*0x80 [x=0..2]

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
	PMUXO[3:0]				PMUXE[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:4 – PMUXO[3:0]: Peripheral Multiplexing Odd**
  These bits select the peripheral function for odd-numbered pins (2*n + 1) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is one.
  Not all possible values for this selection may be valid. For more details, refer to "I/O Multiplexing and Considerations" on page 21.

Table 22-3. Peripheral Multiplexing Odd

PMUXO[3:0]	Name	Description
0x0	A	Peripheral function A selected
0x1	B	Peripheral function B selected
0x2	C	Peripheral function C selected
0x3	D	Peripheral function D selected
0x4	E	Peripheral function E selected
0x5	F	Peripheral function F selected
0x6	G	Peripheral function G selected
0x7	H	Peripheral function H selected
0x8-0xF		Reserved

- **Bits 3:0 – PMUXE[3:0]: Peripheral Multiplexing Even**
  These bits select the peripheral function for even-numbered pins (2*n) of a PORT group, if the corresponding PINCFGy.PMUXEN bit is one.

ADDITIONAL PIN CONFIGURATION

Pin Configuration n

There are up to 32 Pin Configuration registers in each group, one for each I/O line. The n denotes the number of the I/O line, while the x denotes the number of the Port group.

Name: PINCFGn

Offset: 0x40+n*0x1 [n=0..31]+x*0x80 [x=0..2]

Reset: 0x00

Property: Write-Protected

Bit	7	6	5	4	3	2	1	0
		DRVSTR				PULLEN	INEN	PMUXEN
Access	R	R/W	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.
- **Bit 6 – DRVSTR: Output Driver Strength Selection**
  This bit controls the output driver strength of an I/O pin configured as an output.
  0: Pin drive strength is set to normal drive strength.
  1: Pin drive strength is set to stronger drive strength.
- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 2 – PULLEN: Pull Enable**
  This bit enables the internal pull-up or pull-down resistor of an I/O pin configured as an input.
  0: Internal pull resistor is disabled, and the input is in a high-impedance configuration.
  1: Internal pull resistor is enabled, and the input is driven to a defined logic level in the absence of external input.
- **Bit 1 – INEN: Input Enable**
  This bit controls the input buffer of an I/O pin configured as either an input or output.
  0: Input buffer for the I/O pin is disabled, and the input value will not be sampled.
  1: Input buffer for the I/O pin is enabled, and the input value will be sampled when required.
  Writing a zero to this bit disables the input buffer completely, preventing read-back of the physical pin state when the pin is configured as either an input or output.
- **Bit 0 – PMUXEN: Peripheral Multiplexer Enable**
  This bit enables or disables the peripheral multiplexer selection set in the Peripheral Multiplexing register (PMUXn) to enable or disable alternative peripheral control over an I/O pin direction and output drive value.
  0: The peripheral multiplexer selection is disabled, and the PORT registers control the direction and output drive value.
  1: The peripheral multiplexer selection is enabled, and the selected peripheral controls the direction and output drive value.
  Writing a zero to this bit allows the PORT to control the pad direction via the Data Direction register (DIR) and output drive value via the Data Output Value register (OUT). The peripheral multiplexer value in PMUXn is ignored.
  Writing a one to this bit enables the peripheral selection in PMUXn to control the pad. In this configuration, the physical pin state may still be read from the Data Input Value register (IN) if PINCFGy.INEN is set.

USING GPIO

1. Set input/output
2. Set other configuration
3. Read/write

EXAMPLE: USING GPIO

```
void setup() {  // put your setup code here, to run once:
//We know that the LED is at Port 0 pin 5
*REG_PORT_DIR0 |= (1U << 5);       //data direction reg
}
void loop() {  // put your main code here, to run repeatedly
*REG_PORT_OUT0 |= (1U << 5);      //on
delay(1000);
*REG_PORT_OUT 0&= ~(1U << 5);      //off
delay(1000);}
```

EXAMPLE: USING GPIO

```
void setup() {  // put your setup code here, to run once:
//We know that the LED is at Port 0 pin 5
*REG_PORT_DIR0 |= (1U << 5);       //data direction reg
}
void loop() {  // put your main code here, to run repeatedly
*REG_PORT_OUTSET0 |= (1U << 5);      //on
delay(1000);
*REG_PORT_OUTCLR0 |= (1U << 5);     //off
delay(1000);}
```

EXAMPLE: USING GPIO

```
void setup() {  // put your setup code here, to run once:
//We know that the LED is at Port 0 pin 5
*REG_PORT_DIR0 |= (1U << 5);       //data direction reg
}
void loop() {  // put your main code here, to run repeatedly
*REG_PORT_OUTTGL0 ^= (1U << 5);      //Toggle
delay(1000);
```

SPECIFIC PERIPHERAL CONCEPTS

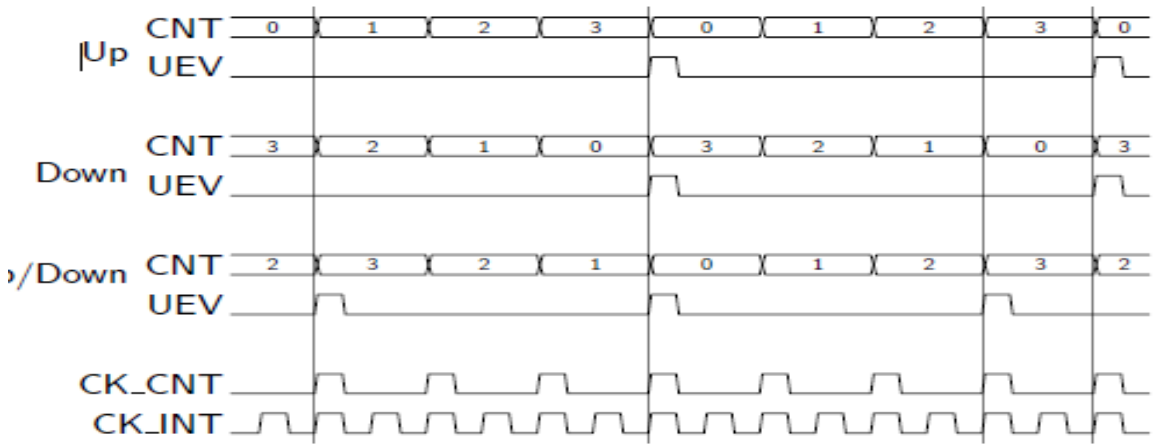
ARM Timers

Definition?

Timers are **hardware Counters** that can be configured to run from a variety of internal or external clocks.

They run *concurrently* with CPU and can be configured in software to provide a variety of functions.

Counter Modes (Example)

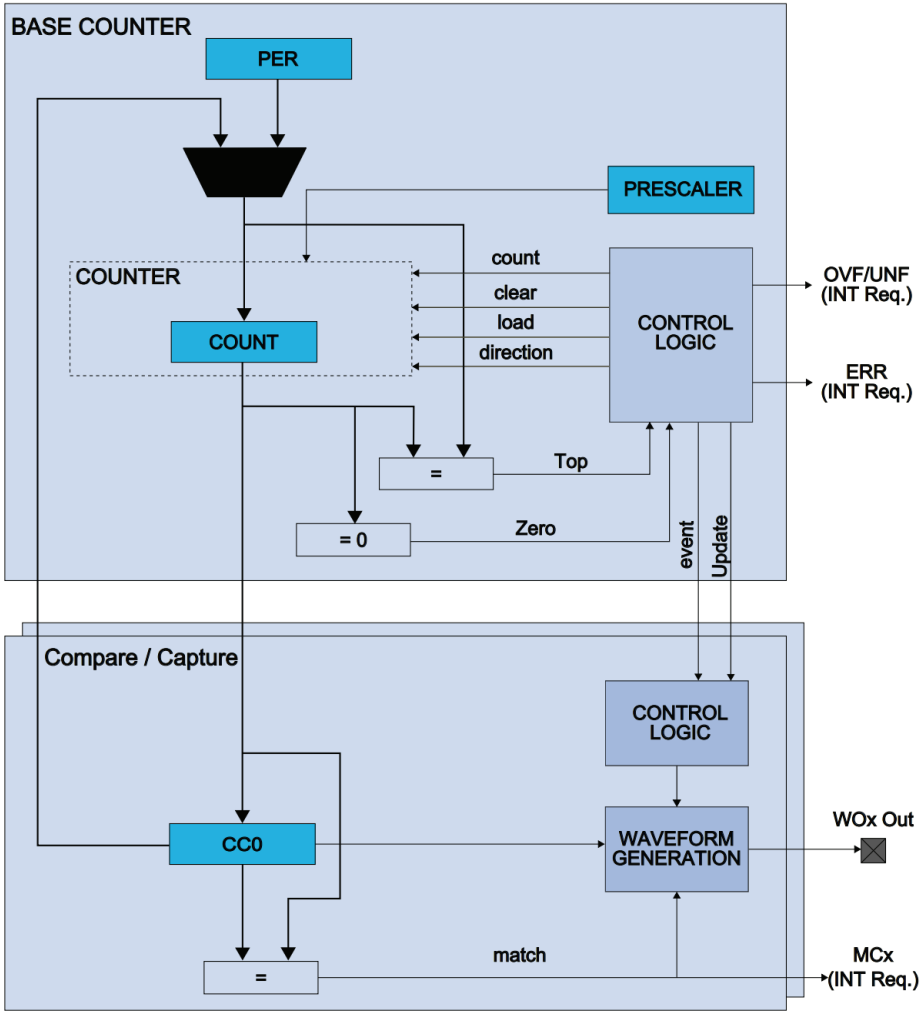


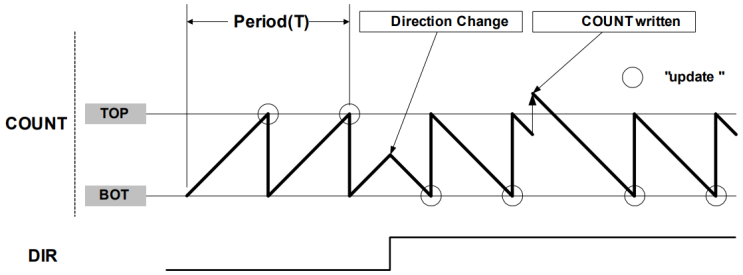
Counter Modes (ARR=3, PSC=1)

What about our Mo+?

- Selectable configuration
  - 8-, 16- or 32-bit TC, with compare/capture channels
- Waveform generation
  - Frequency generation
  - Single-slope pulse-width modulation
- Input capture
  - Event capture
  - Frequency capture
  - Pulse-width capture
- One input event
- Interrupts/output events on:
  - Counter overflow/underflow
  - Compare match or capture
- Internal prescaler
- Can be used with DMA and to trigger DMA transactions

Timer Block Diagram





Example: LPC2148

- 2 x 32 Bit Identical Timer blocks
  Either Timer or Counter
- 2 Important Registers:
  TC (Timer Counter)
  PR (Prescaler)
- If timer is reset and enabled:
  1. TC->0
  2. TC++, every PR+1 clock ticks
  3. TC resets to zero at 0xFFFF FFFF
- 4 Match Registers and 4 Capture Registers for each Timer
  Do I have a match??
  Stop the Timer, reset the Timer, generate an interrupt, or do whatever else the Match Control Register is configured for….

What about Capture?
Configure a pin to store current counter value in a capture register.

Example: LPC2148

- ***Prescale Register (PR)***:

Prescale Register specifies the maximum value for the Prescale Counter.
When the Prescale Counter (PC) is equal to PR, the TC in incremented on the next clock and also PC is cleared.

- ***Prescale Counter Register (PC)***:

Prescale Counter increments on every peripheral clock to the value stored in the PR.
When the value in the PC is equal to the value in the PR, PC is reset and TC is incremented by 1 on the next clock cycle.
For example, if PR = 2, then TC is incremented on every third cycle of the peripheral clock.

Example: LPC2148

- ***Timer Control Register (TCR)***:

Timer Control Register is used to control the functions of timer / counter.
It is used to enable / disable or reset the Timer Counter.
Bit 0 = 0, Timer Counter and Prescale Counter are disabled.
Bit 0 = 1, the counters are enabled.
Bit 1 = 1, both the counters (Timer Counter and Prescale Counter) are reset on the next positive edge of the peripheral clock. They remain reset until Bit 1 = 0.

- ***Count Timer Control Register (CTCR)***:

Sets Timer/Counter Mode.
If Counter Mode is selected the counter pin and the
edges (rising, falling or both) can be selected using CTCR.

Example: LPC2148

Match Control Register (MCR):
The Match Control Register is used control the actions to be performed when the value in the Match Register (MR) matches with the value in the Timer Counter (TC).
.
Bit 0: When this bit is 1, an interrupt is triggered when MR0 is equal to TC.
When this bit is 0, the interrupts is disabled.
Bit 1: When this bit is 1, TC is reset when MR0 is equal to TC.
When this bit is 0, this feature is disabled.
Bit 2: When this bit is 1, the Timer Counter (TC) and Prescale Counter (PC) are stopped when the value in MR0 is equal to TC. Also, the TC is reset to 0.

Bit 3-5, Used for MR1 etc…

Example: LPC2148

If the clock = 60 MHz

```
T0CTCR = 0x0; // Timer Mode is selected.
T0PR = 59999; // PR is set to this value
T0TCR = 0x02; // Reset the Timer
```

How often will TC be incremented?
Is the Timer running?

Example: LPC2148

No!

```
T0TCR = 0x01; // Enable the Timer
while (T0TC < time_in_milliseconds); //Wait till the TC
reaches the desired delay
T0TCR = 0x00; // Disable the Timer
```

Example: LPC2148

```
void delay (int d)
{
T0TCR=0x02;
T0TCR=0x01;
while (T0TC < d);
T0TCR=0x00;
}
int main()
{
 .....
/* Initialize the Timer */
T0CTCR=0x00;
T0PR=59999;
T0TCR=0x02;
.....
}
```

Example:Timer3 Toggles LED at One Second intervals.

```
// Assume CPU configured to run at system clock frequency 24 MHz.

// GPIO driving LED has been configured for output and its clock enabled.

#include "stm32f10x.h“   // Contains definitions of Timer structure


    RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;  //  enable clock for timer TIM3 :

     TIM3->PSC = 23999;                    // Set pre-scaler to 24 000 (PSC + 1)
      TIM3->ARR = 1000;                     // Auto reload value 1000
      TIM3->CR1 = TIM_CR1_CEN;              // Enable timer
```

Example:Timer3 Toggles LED at One Second intervals.

```
// When timer reaches the ARR value, the UIF  flag will be set. Check this flag, clear it

// and toggle an LED :

    while(1){

        if(TIM3->SR & TIM_SR_UIF)
                {
                   TIM3->SR &= ~TIM_SR_UIF;
                   LED_BLUE_GPIO->ODR ^= (1 << LED_BLUE_PIN);
                }

     }
```

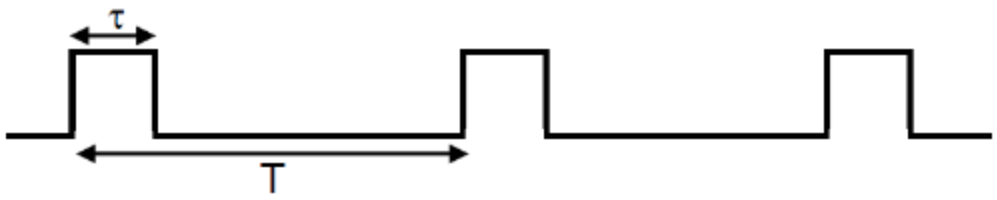
Capture and Compare Register (CCR)

CCR is based on a Timer Register plus additional hardware.
CCR can be configured for either Capture or Compare modes:

- Capture mode operates as a stop watch
  - – You start the stop watch
  - – The stop watch runs while you wait for an event(s) to happen
  - – Once the event(s) happen the stop watch is stopped and the time is recorded

- Compare mode operates as a comparator
  - -You set a predefined value
  - - Start the register to count incoming pulses
  - - Once the pre-defined value is reached it generates an output event

Pulse Width Modulation (PWM)



τ – Pulse width
T – Period
Duty cycle = (τ/T)*100%

Example:
τ = 1μsec, T = 20μsec then duty cycle is 5%

PWM (PULSE WIDTH MODULATION)

PWM is based on switching between a small number of voltage levels (typically two voltage levels v_{min} and v_{max}, e.g., 0 V and 5 V).

- The periods of time that the signal takes the values v_{min} and v_{max} within a pulse period are called the OFF period and ON period, respectively. If these time periods are denoted as T_{off} and T_{on}, respectively, the corresponding average value is $\frac{T_{on}v_{max}+T_{off}v_{min}}{T_{on}+T_{off}}$.
- The duty cycle is defined as $\frac{T_{on}}{T_{on}+T_{off}}$.

APPROXIMATE CONTINUOUS RANGE

Given a desired analog voltage v_{avg}, a PWM approximation for the signal can be defined by picking T_{on} and T_{off} such that $\frac{T_{on}v_{max}+T_{off}v_{min}}{T_{on}+T_{off}} = v_{avg}$

- If $v_{min} = 0$, then this equation simplifies to $\frac{T_{on}}{T_{on}+T_{off}}v_{max} = v_{avg}$.
- The sum $T_{on} + T_{off}$ is often picked to be a constant and the parts T_{on} and T_{off} are picked based on the desired v_{avg}. Hence, if the sum $(T_{on} + T_{off})$ is specified as a constant T, then T_{on} and T_{off} can be found using

$$T_{on} = T\frac{v_{avg} - v_{min}}{v_{max} - v_{min}}$$

$$T_{off} = T - T_{on}$$

If $v_{min} = 0$, then we simply get $T_{on} = T\frac{v_{avg}}{v_{max}}$

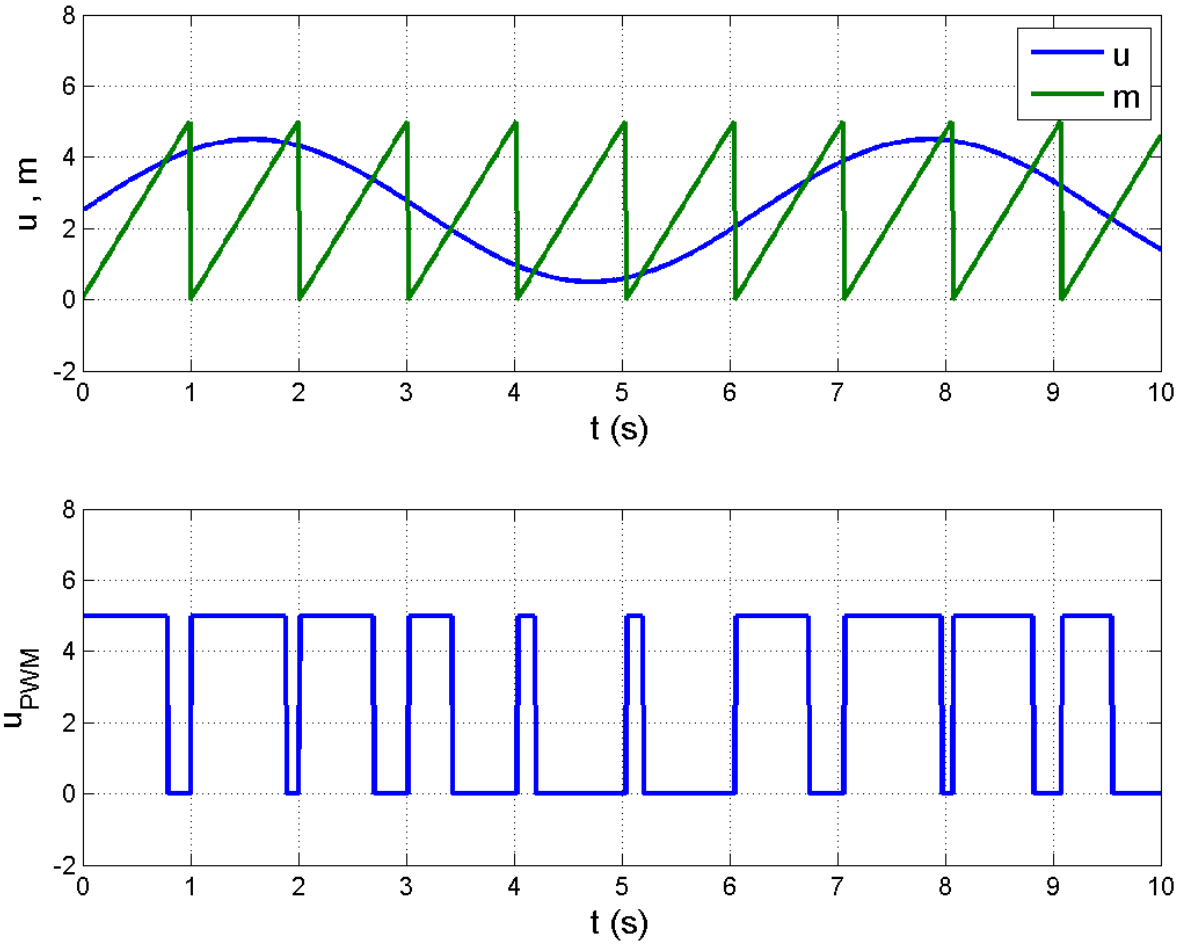
INTERSECTIVE PWM METHOD

In the intersective PWM method, a sawtooth waveform m(t) is generated and the signal u(t) is compared (at each time instant) with the sawtooth waveform m(t). The output PWM signal $u_{PWM}(t)$ is defined as follows:

If $u(t) > m(t)$ at a time t, then at that time, $u_{PWM} = v_{max}$.

If $u(t) < m(t)$ at a time t, then at that time, $u_{PWM} = v_{min}$.

ILLUSTRATION OF INTERSECTIVE PWM METHOD



Example of intersective PWM method. Approximation of an analog signal $u(t) = 2.5 + 2 \sin(t)$ using a PWM signal with $v_{min} = 0$ V and $v_{max} = 5$ V. Here, the time step dt is set to 0.01 seconds and the PWM period T is set to 1 second.

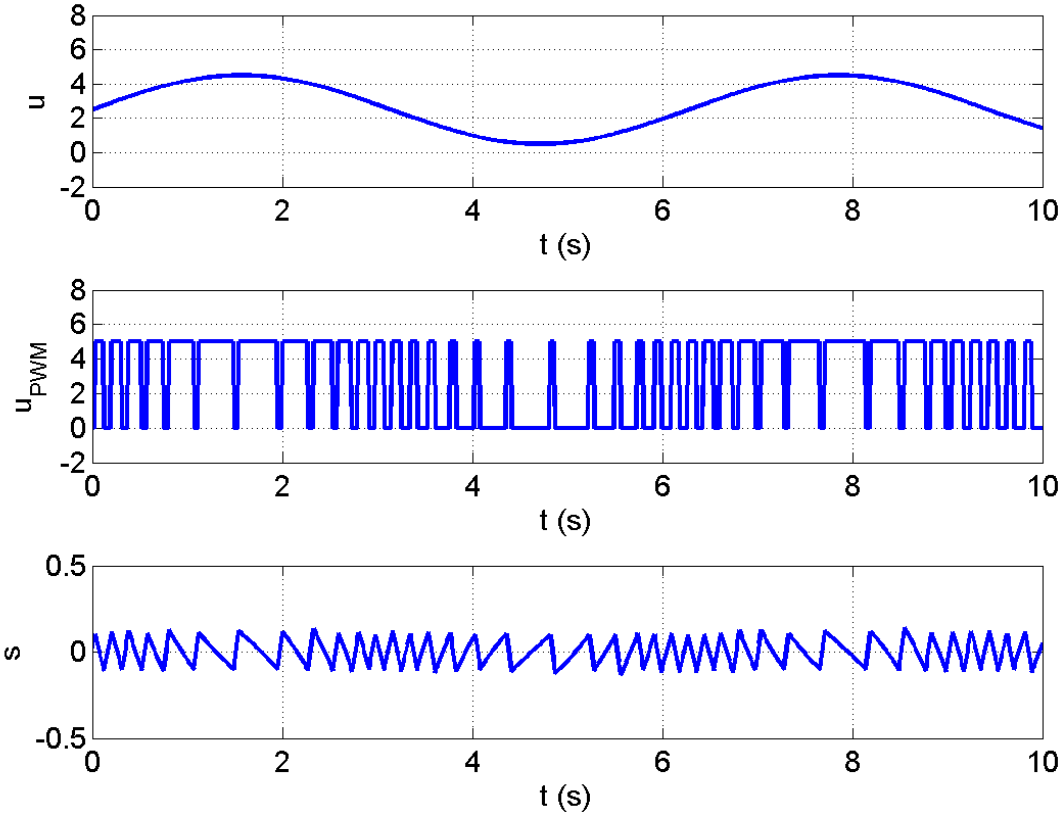
DELTA PWM METHOD

In the delta PWM method, a running integral of the deviation between the analog input signal u(t) and the PWM output signal $u_{PWM}(t)$ is maintained. When the integral crosses a threshold (i.e., when the integral becomes too big in magnitude, either as a positive number or a negative number), then the output u_{PWM} is switched.

Defining the integral $s(t) = \int_0^t (u(\tau) - u_{PWM}(\tau))d\tau$, we define the output signal switching to be as follows:

- If $s(t) > \epsilon$, then switch u_{PWM} to v_{max}
- If $s(t) < -\epsilon$, then switch u_{PWM} to v_{min}

ILLUSTRATION OF DELTA PWM METHOD



Example of delta PWM method. Approximation of an analog signal $u(t) = 2.5 + 2 \sin(t)$ using a PWM signal with $v_{min} = 0$ V and $v_{max} = 5$ V. Here, the time step dt is set to 0.01 seconds and the threshold ϵ is set to 0.1.

ILLUSTRATION OF PWM METHOD on M0+

- CCx registers control the duty cycle

- COUNT UP:

WO[x] output is set at a start or compare match between the COUNT value and the top value and cleared on the compare match between the COUNT value and CCx register value.

- COUNT DOWN:

WO[x] output is cleared at start or compare match between the COUNT value and the top value and set on the compare match between the COUNT value and CCx register value.

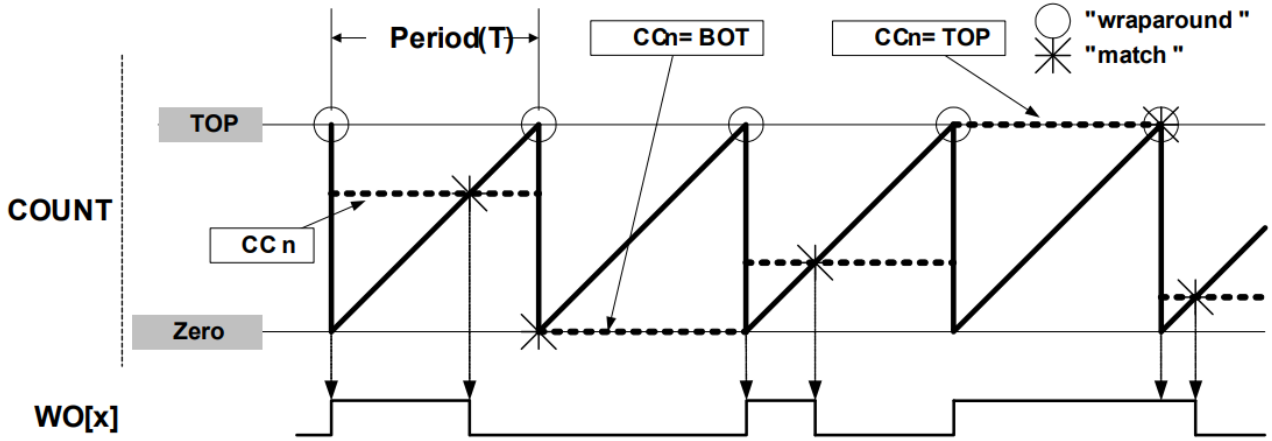
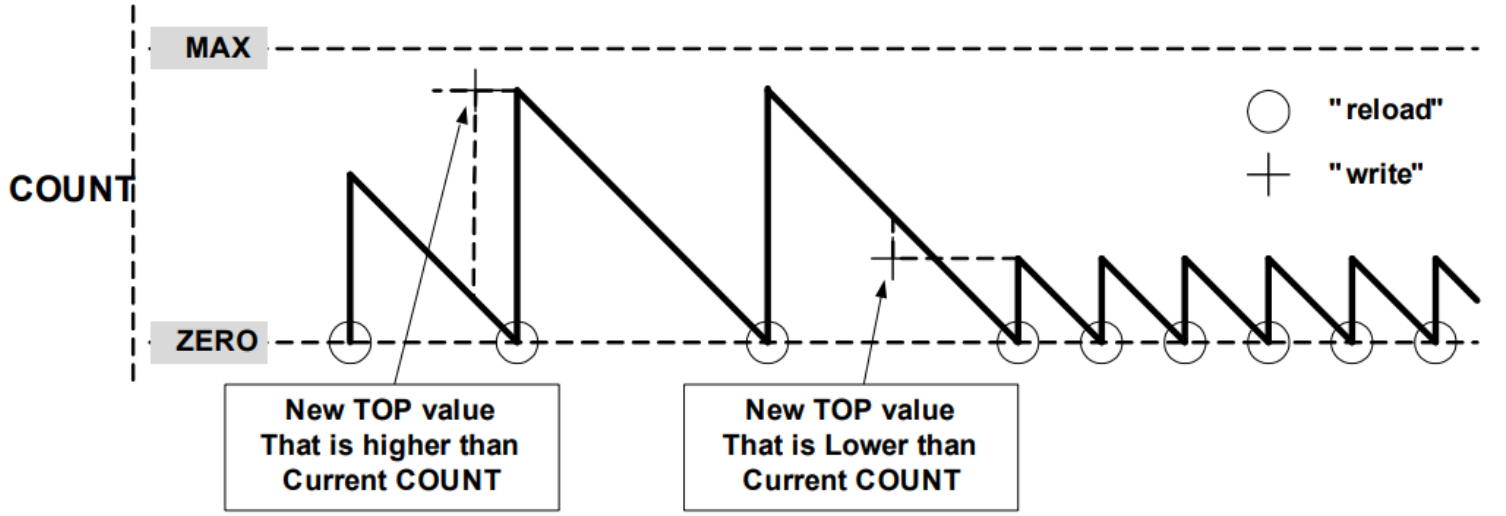
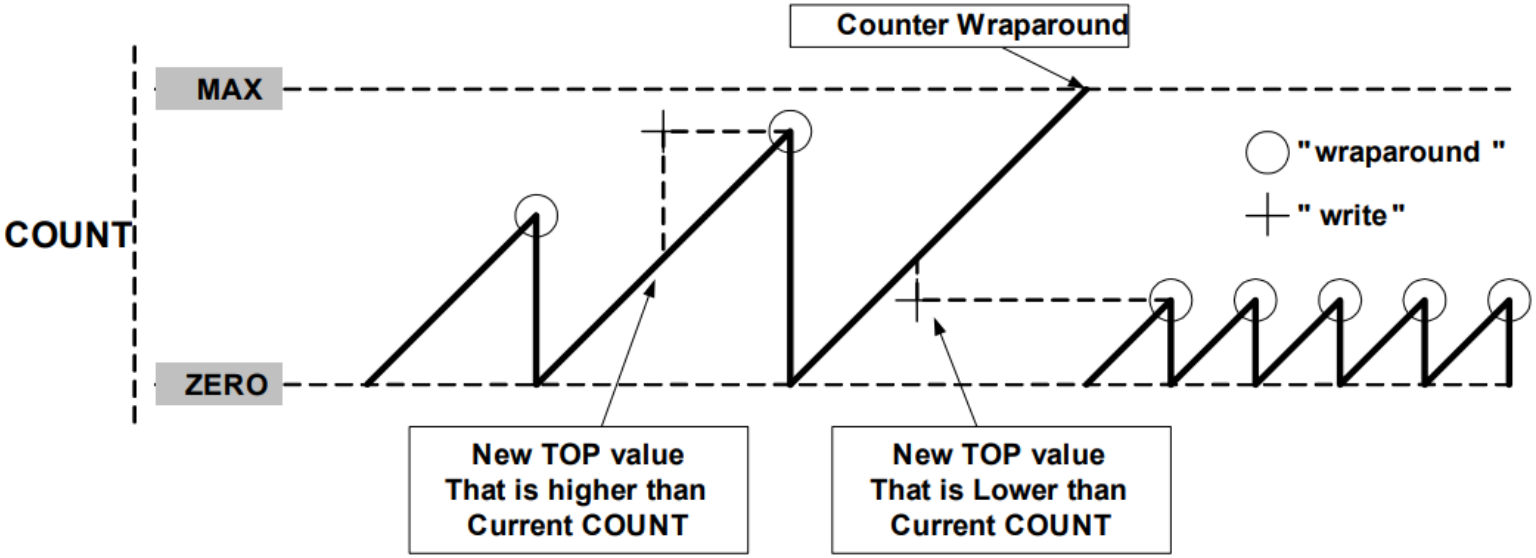


ILLUSTRATION OF PWM METHOD on M0+ - TOP Change





Registers For Timers (8-Bit, similar for 16 and 32)

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		WAVEGEN[1:0]			MODE[1:0]		ENABLE	SWRST
0x01		15:8			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
0x02	READREQ	7:0				ADDR[4:0]				
0x03		15:8	RREQ	RCONT						
0x04	CTRLBCLR	7:0	CMD[1:0]					ONESHOT		DIR
0x05	CTRLBSET	7:0	CMD[1:0]					ONESHOT		DIR
0x06	CTRLC	7:0			CPTEN1	CPTEN0			INVEN1	INVEN0
0x07	Reserved									
0x08	DBGCTRL	7:0								DBGRUN
0x09	Reserved									
0x0A	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
0x0B		15:8			MCEO1	MCEO0				OVFEO
0x0C	INTENCLR	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0D	INTENSET	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0E	INTFLAG	7:0			MC1	MC0	SYNCRDY		ERR	OVF
0x0F	STATUS	7:0	SYNCBUSY			SLAVE	STOP			
0x10	COUNT	7:0	COUNT[7:0]							
0x11	Reserved									
0x12	Reserved									
0x13	Reserved									
0x14	PER	7:0	PER[7:0]							
0x15	Reserved									
0x16	Reserved									
0x17	Reserved									
0x18	CC0	7:0	CC[7:0]							
0x19	CC1	7:0	CC[7:0]							

Registers For Timers (8-Bit, similar for 16 and 32)

29.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x0000
Property: Write-Protected, Enable-Protected, Write-Synchronized

Bit	15	14	13	12	11	10	9	8
			PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		WAVEGEN[1:0]			MODE[1:0]		ENABLE	SWRST
Access	R	R/W	R/W	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 13:12 – PRESCSYNC[1:0]: Prescaler and Counter Synchronization**
  These bits select whether on start or retrigger event the counter should wrap around on the next GCLK or the next prescaled GCLK_TCx clock. It's also possible to reset the prescaler.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

- **Bit 11 – RUNSTDBY: Run in Standby**
  This bit is used to keep the TC running in standby mode:
  0: The TC is halted in standby.
  1: The TC continues to run in standby.
  This bit is not synchronized.

- **Bits 10:8 – PRESCALER[2:0]: Prescaler**
  These bits select the counter prescaler factor, as shown in Table 29-6.
  These bits are not synchronized.

Table 29-6. Prescaler

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

- **Bit 7 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 6:5 – WAVEGEN[1:0]: Waveform Generation Operation**
  These bits select the waveform generation operation. They affect the top value, as shown in "Waveform Output Operations" on page 616. It also controls whether frequency or PWM waveform generation should be used. How these modes differ can also be seen from "Waveform Output Operations" on page 616.
  These bits are not synchronized.

Table 29-7. Waveform Generation Operation

Value	Name	Operation	Top Value	Waveform Output on Match	Waveform Output on Wraparound
0x0	NFRQ	Normal frequency	PER(1)/Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER(1)/Max	Clear when counting up Set when counting down	Set when counting up Clear when counting down
0x3	MPWM	Match PWM	CC0	Clear when counting up Set when counting down	Set when counting up Clear when counting down

Note: 1. This depends on the TC mode. In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode it is the maximum value.

- **Bit 4 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written. This bit will always return zero when read.

- **Bits 3:2 – MODE[1:0]: TC Mode**
  These bits select the TC mode, as shown in Table 29-8.
  These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
0x3	-	Reserved

Registers For Timers (8-Bit, similar for 16 and 32)

29.8.4 Control B Set

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBCLR) register.

Name: CTRLBSET
Offset: 0x05
Reset: 0x00
Property: Write-Protected, Write-Synchronized, Read-Synchronized

Bit	7	6	5	4	3	2	1	0
	CMD[1:0]					ONESHOT		DIR
Access	R/W	R/W	R	R	R	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – CMD[1:0]: Command**
  These bits is used for software control of retriggering and stopping the TC. When a command has been executed, the CMD bit group will be read back as zero. The commands are executed on the next prescaled GCLK_TC clock cycle.
  Writing a zero to one of these bits has no effect.
  Writing a one to one of these bits will set a command.

Table 29-10. Command

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	-	Reserved

- **Bits 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bit 2 – ONESHOT: One-Shot**
  This bit controls one-shot operation of the TC. When active, the TC will stop counting on the next overflow/underflow condition or a stop command.
  0: The TC will wrap around and continue counting on an overflow/underflow condition.
  1: The timer/counter will wrap around and stop on the next underflow/overflow condition.
  Writing a zero to this bit has no effect.
  Writing a one to this bit will enable one-shot operation.

29.8.5 Control C

Name: CTRLC
Offset: 0x06
Reset: 0x00
Property: Write-Protected, Write-Synchronized, Read-Synchronized

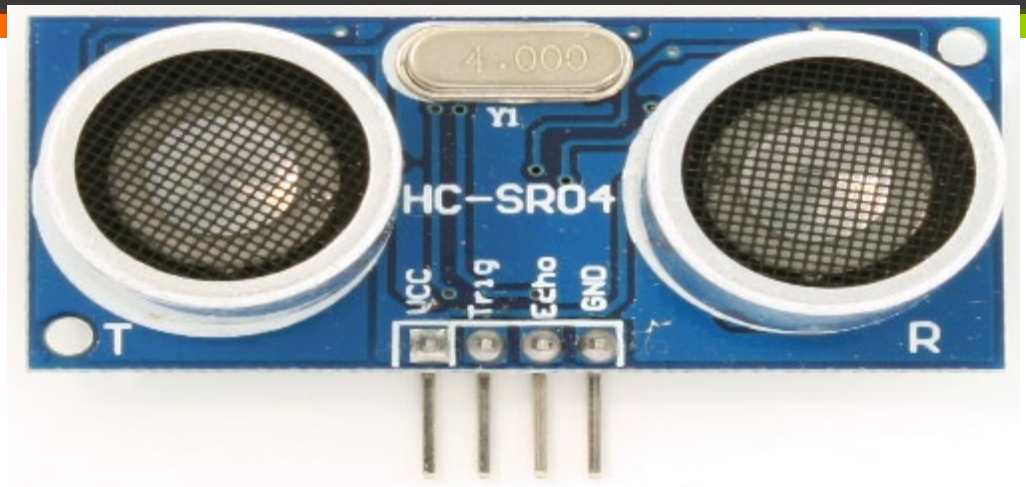
Bit	7	6	5	4	3	2	1	0
			CPTEN1	CPTEN0			INVEN1	INVEN0
Access	R	R	R/W	R/W	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- **Bits 7:6 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 5:4 – CPTENx: Capture Channel x Enable**
  These bits are used to select whether channel x is a capture or a compare channel.
  Writing a one to CPTENx enables capture on channel x.
  Writing a zero to CPTENx disables capture on channel x.
- **Bits 3:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written. These bits will always return zero when read.
- **Bits 1:0 – INVENx: Waveform Output x Invert Enable**
  These bits are used to select inversion on the output of channel x.
  Writing a one to INVENx inverts the output from WO[x].
  Writing a zero to INVENx disables inversion of the output from WO[x].

Registers For Timers (as per the HAL)

```
/* ========== Register definition for TC3 peripheral ========== */
#if (defined(__ASSEMBLY__) || defined(__IAR_SYSTEMS_ASM__))
#define REG_TC3_CTRLA              (0x42002C00U) /**< \brief (TC3) Control A */
#define REG_TC3_READREQ            (0x42002C02U) /**< \brief (TC3) Read Request */
#define REG_TC3_CTRLBCLR           (0x42002C04U) /**< \brief (TC3) Control B Clear */
#define REG_TC3_CTRLBSET           (0x42002C05U) /**< \brief (TC3) Control B Set */
#define REG_TC3_CTRLC              (0x42002C06U) /**< \brief (TC3) Control C */
#define REG_TC3_DBGCTRL            (0x42002C08U) /**< \brief (TC3) Debug Control */
#define REG_TC3_EVCTRL             (0x42002C0AU) /**< \brief (TC3) Event Control */
#define REG_TC3_INTENCLR           (0x42002C0CU) /**< \brief (TC3) Interrupt Enable Clear */
#define REG_TC3_INTENSET           (0x42002C0DU) /**< \brief (TC3) Interrupt Enable Set */
#define REG_TC3_INTFLAG            (0x42002C0EU) /**< \brief (TC3) Interrupt Flag Status and Clear
#define REG_TC3_STATUS             (0x42002C0FU) /**< \brief (TC3) Status */
#define REG_TC3_COUNT16_COUNT      (0x42002C10U) /**< \brief (TC3) COUNT16 Counter Value */
#define REG_TC3_COUNT16_CC0        (0x42002C18U) /**< \brief (TC3) COUNT16 Compare/Capture 0 */
#define REG_TC3_COUNT16_CC1        (0x42002C1AU) /**< \brief (TC3) COUNT16 Compare/Capture 1 */
#define REG_TC3_COUNT32_COUNT      (0x42002C10U) /**< \brief (TC3) COUNT32 Counter Value */
#define REG_TC3_COUNT32_CC0        (0x42002C18U) /**< \brief (TC3) COUNT32 Compare/Capture 0 */
#define REG_TC3_COUNT32_CC1        (0x42002C1CU) /**< \brief (TC3) COUNT32 Compare/Capture 1 */
#define REG_TC3_COUNT8_COUNT       (0x42002C10U) /**< \brief (TC3) COUNT8 Counter Value */
#define REG_TC3_COUNT8_PER         (0x42002C14U) /**< \brief (TC3) COUNT8 Period Value */
#define REG_TC3_COUNT8_CC0         (0x42002C18U) /**< \brief (TC3) COUNT8 Compare/Capture 0 */
#define REG_TC3_COUNT8_CC1         (0x42002C19U) /**< \brief (TC3) COUNT8 Compare/Capture 1 */
#else
```

Input Capture: Ultrasonic Ranger

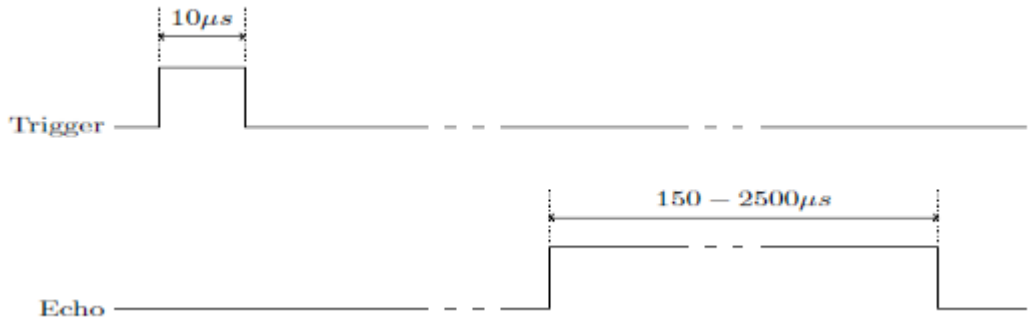


HC-SR04 ultrasonic module:

- 3mm resolution in the range 20-500 mm
- Ultrasonic Pulse triggered by delivering to it a 10 us pulse
- Generates an 8 pulse 40 kHz signal driving the Ultrasonic transducer
- Distance determined as a Pulse Width obtained from PWM module output.

distance = pulse width * 58×10^{-6} [cm]

Ultrasonic Sensor Protocol



Implementation of Measurement:

Use 2 Timers.

- TIM4 to generate the trigger pulse.
- TIM1 to measure the echo pulse.

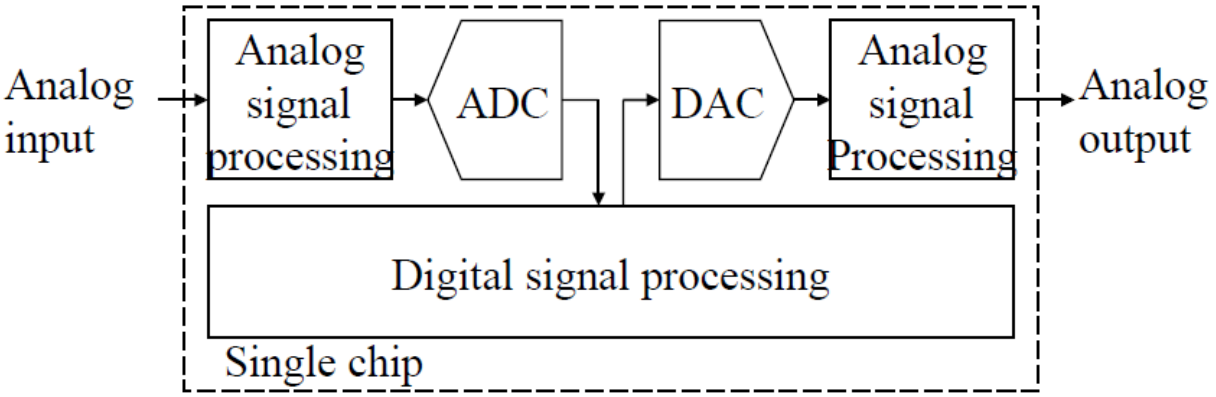
Outline of Algorithm

1. Configure TIM1 prescaler and period
2. Configure TIM1 channel 1 to latch the timer value on rising input
3. Configure TIM1 channel 2 to latch timer value on a falling input
4. Configure TIM1 to reset on the capture of completed event
5. Enable Interrupt for complete TIM1 event
6. Write ISR to store the difference between falling and rising edges and signal data ready by setting a user defined flag.

A/D AND D/A CONVERSION

- An ADC is used to read an analog signal (a voltage or a current) into a digital computer, e.g., to read an analog sensor.
- A DAC is used to write out an analog signal (a voltage or a current) from a digital computer, e.g., outputting a voltage to control an electric motor.

A/D AND D/A CONVERSION



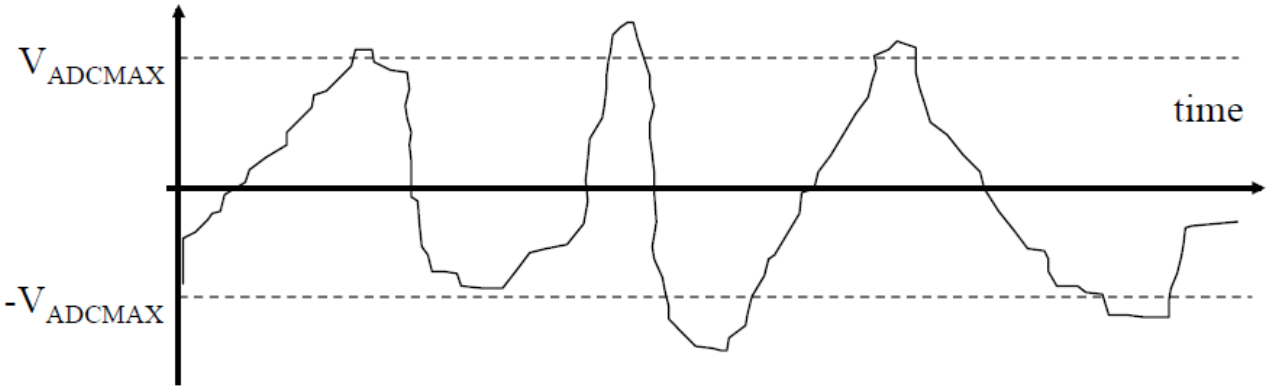
- Signals are analog by nature
- ADC necessary for DSP
- Digital signal processing provides:
  - Close to infinite SNR
  - Low system cost
  - Repetitive system

- ADC bottle necks:
  - Dynamic range
  - Conversion speed
  - Power consumption

A/D CONVERSION

- Input Range
  - Unipolar: (0, V_{ADCMAX})
  - Bipolar: ($-V_{ADCMAX}$, $+V_{ADCMAX}$) (Nominal Range)
  - **Clipping**:

    If $|V_{IN}| > |V_{ADCMAX}|$, then $|V_{OUT}| = |V_{ADCMAX}|$



ADC CHARACTERISTICS

The input signal is an analog signal (voltage, current), and the output is a binary number.

Precision: number of distinguishable ADC inputs (e.g., 12 bits, 2^n -1 = 4095 alternatives)

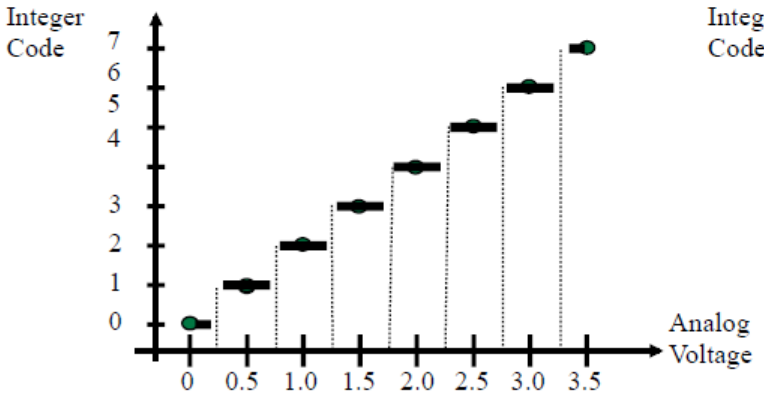
Range: maximum and minimum ADC input (e.g., 0 to +3.3V unipolar).

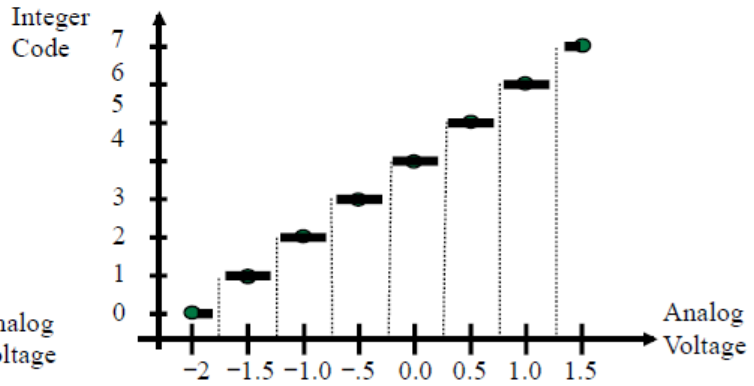
Resolution (Quantization Interval Q): smallest distinguishable change in input (e.g., 3.3V/4096, which is about 0.81 mV). The resolution is the change in input that causes the digital output to change by 1.

ADC EQUATIONS

- Quantization Interval (Q)
  - n bit ADC, the input range is divided into 2^n-1 intervals.
  - 3 bit ADC:

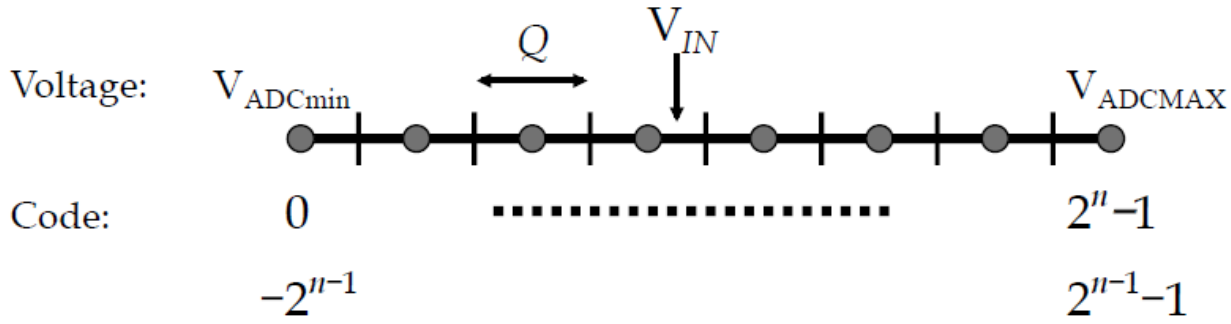
$$Q = \frac{V_{ADCMAX} - V_{ADC\min}}{2^n - 1}$$





ADC EQUATIONS

- **Voltage to Integer Code**
  - *n* bit ADC



Positive Coding:

$$\text{Code} = \text{Round}\left[\frac{V_{IN} - V_{ADC\,min}}{Q}\right]$$

Positive and Negative Coding:

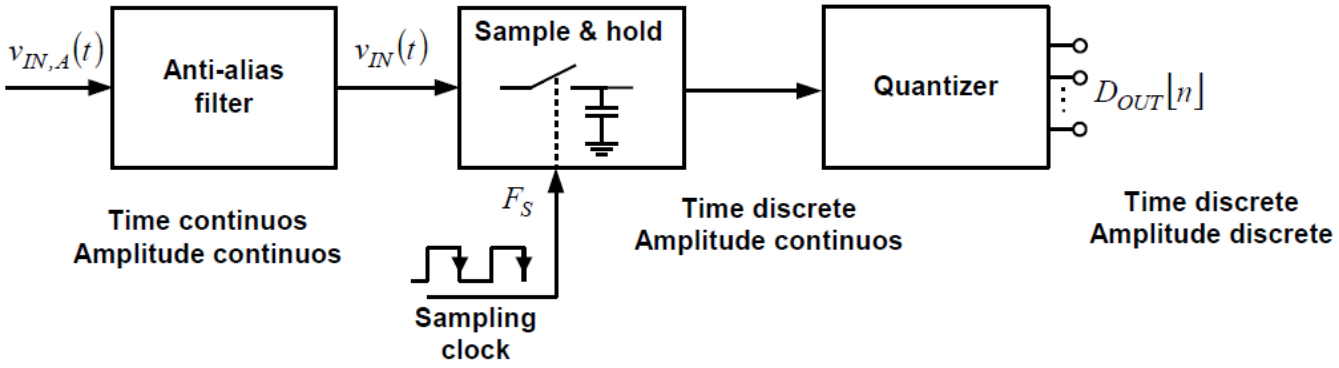
$$\text{Code} = \text{Round}\left[\frac{V_{IN}}{Q}\right]$$

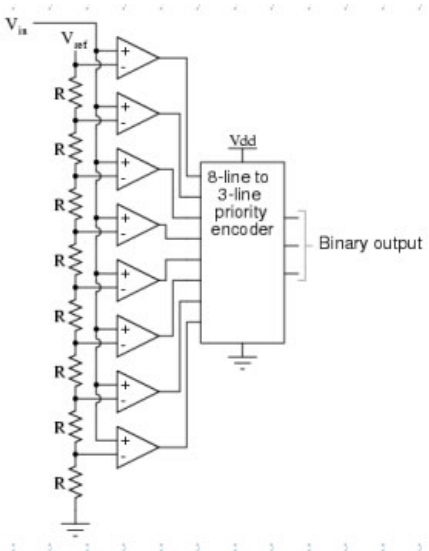
ADC EXAMPLE

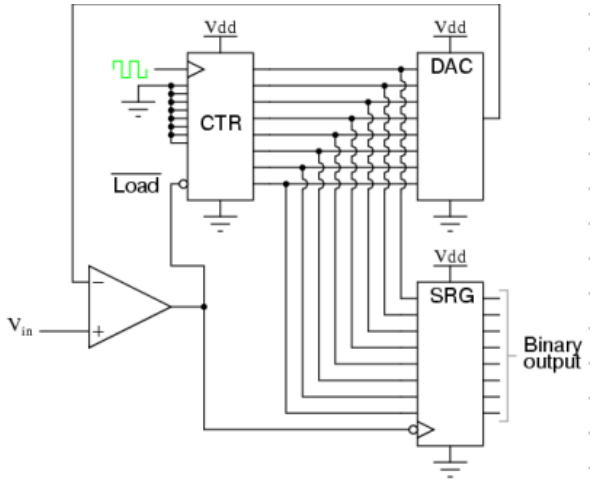
Example: If the input voltage range is 0 – 3.3 V and the number of bits is 12, then if the input voltage is 2.1 V, the corresponding digital value from the ADC is $x = \lfloor (2^{12} - 1)\frac{2.1-0}{3.3-0} \rfloor = 2605$.

ADC IMPLEMENTATION

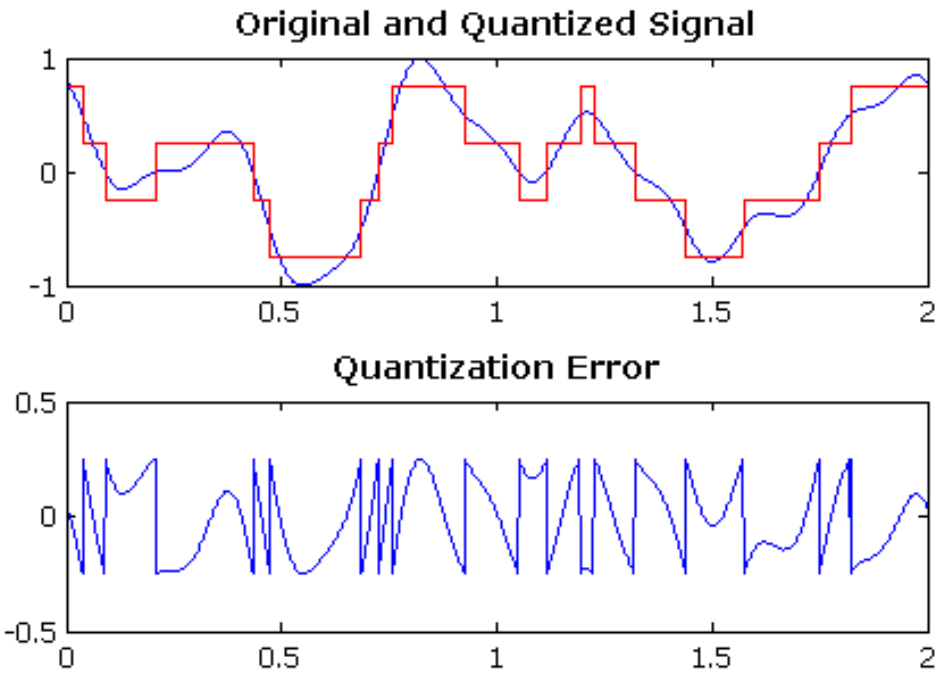
Parallel vs. Ramp vs. Sample and Hold







QUANTIZATION ERROR



QUANTIZATION ERROR

- Noise energy:

$$V_{Q(RMS)} = \sqrt{\frac{1}{\delta}\int_{-\delta/2}^{\delta/2} V_Q^2 dV_Q} = \sqrt{\frac{\delta^2}{12}}$$

- Signal energy:

$$V_{in(RMS)} = \frac{\delta \cdot 2^N}{2\sqrt{2}}$$

- SNR for ideal ADC:

$$SNR = 20\log\left(\frac{V_{in(RMS)}}{V_{Q(RMS)}}\right)$$

$$SNR = 20\log\left(2^N \cdot \sqrt{\frac{3}{2}}\right)$$

$$SNR = 6.02 \times N + 1.76 [dB]$$

EXERCISE

We must measure an input signal that varies between 0mV and 600mV. The system ADC can accept input in the range of 0V to 2.5V.

If the signal is amplified with a gain of 4 before reaching ADC, how many bits of resolution do we need to know the value of the input with max error 0.1%?

How many bits of resolution would you need if you were not using the amplifier?

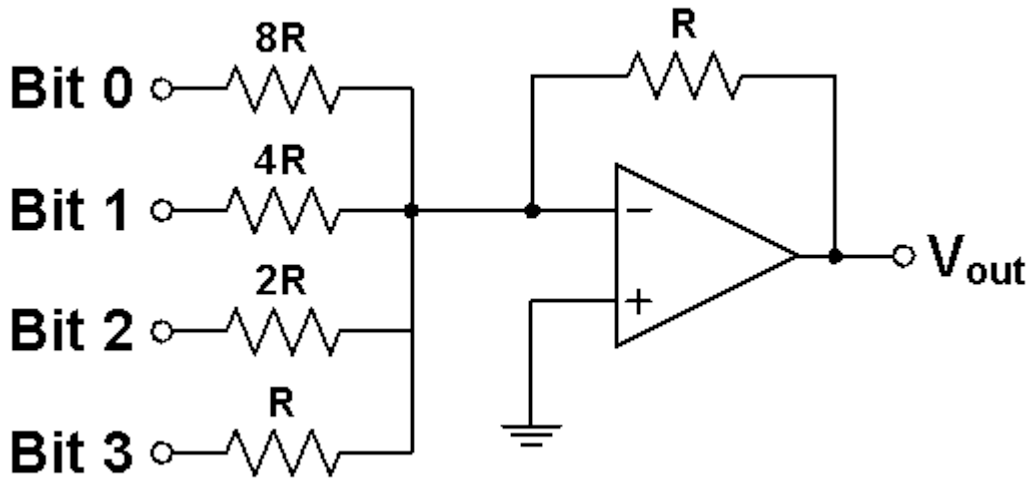
DAC EQUATIONS

If the output voltage range of the DAC is v_{min} to v_{max} and the number of bits is n, then if the digital value (an integer between 0 and $2^n - 1$) written to the DAC is x, the corresponding analog value that will be output by the DAC is:

- $v = x\frac{v_{max} - v_{min}}{N} + v_{min}$ where $N = (2^n - 1)$. Alternatively, $N = 2^n$ is also used in some DAC devices. On the STM32 microcontroller on the STM32F4 Discovery board, $N = 2^n - 1$ is used.

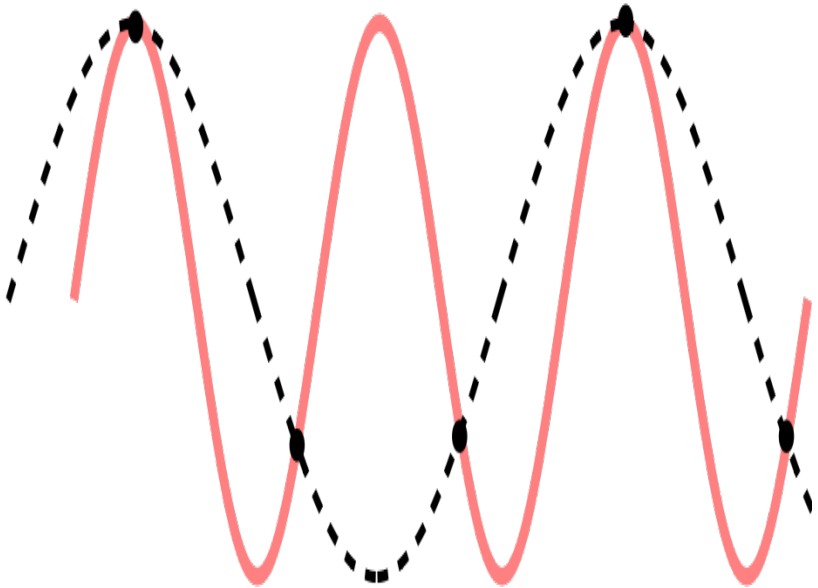
Example: If the output voltage range is 0 – 3.3 V and the number of bits is 12, then if the digital value written to the DAC is 1500, the corresponding analog value that will be output by the DAC is $v = 1500\frac{3.3-0}{2^{12}-1} + 0 = 1.2088V$.

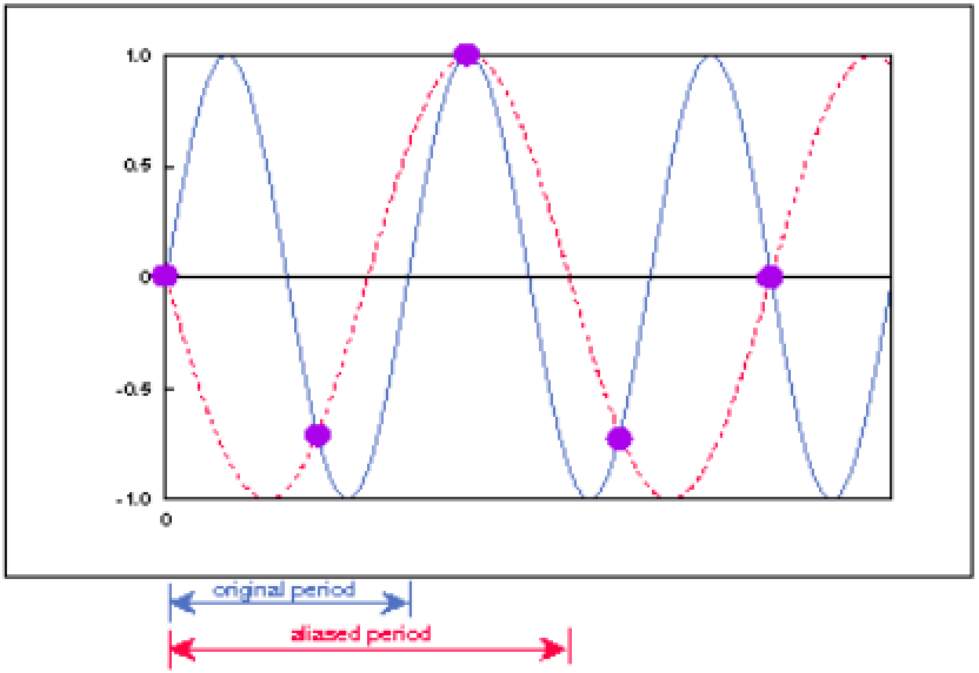
DAC IMPLEMENTATION



SAMPLING

Nyquist theorem: To capture signal that varies at frequency f, you must sample at a rate $f_s > 2f$. See the demo.

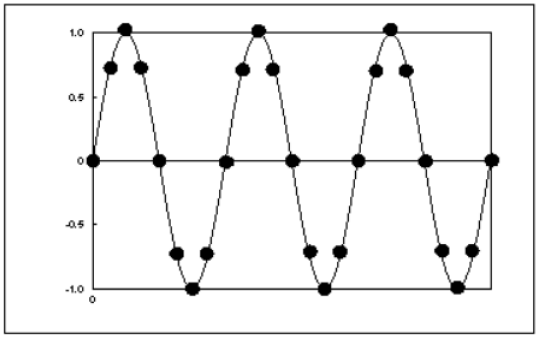




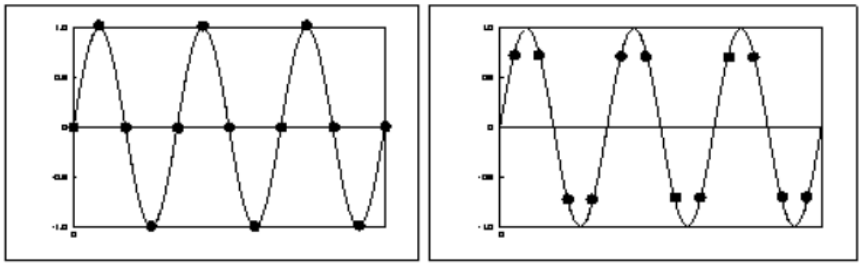
SAMPLING EXAMPLES (NO ALIASING)

Graphical Example 1a:

- SR = 20,000 Hz
- Nyquist Frequency = 10,000 Hz
- f = 2,500 Hz (no aliasing)



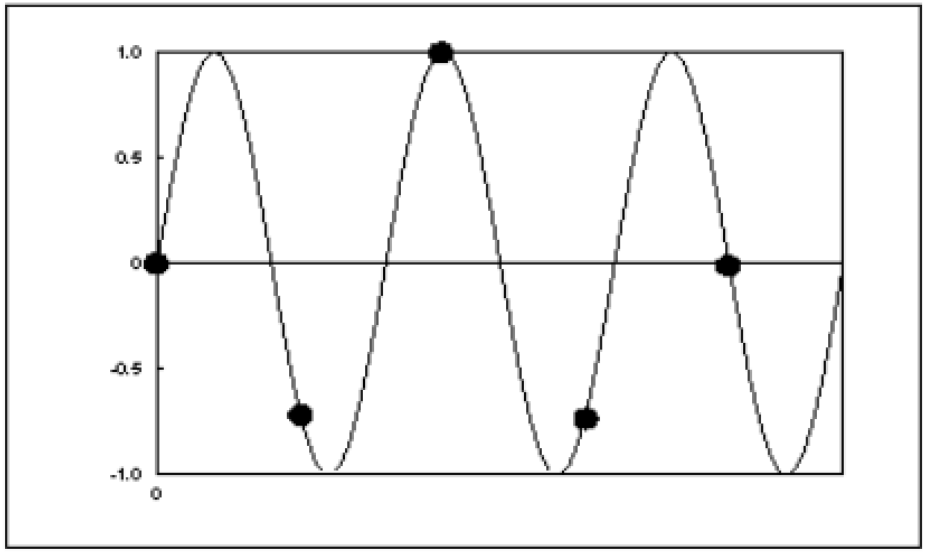
- Graphical Example 1b:
  - SR = 20,000 Hz
  - Nyquist Frequency = 10,000 Hz
  - f = 5,000 Hz (no aliasing)



SAMPLING EXAMPLES (ALIASING)

Graphical Example 3:

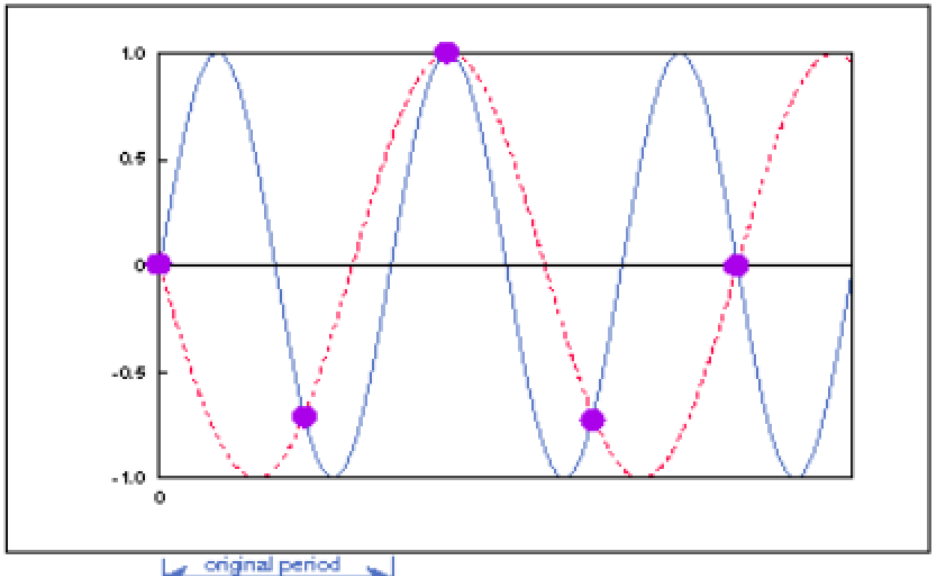
- SR = 20,000 Hz
- Nyquist Frequency = 10,000 Hz
- f = 12,500 Hz, f' = 7,500



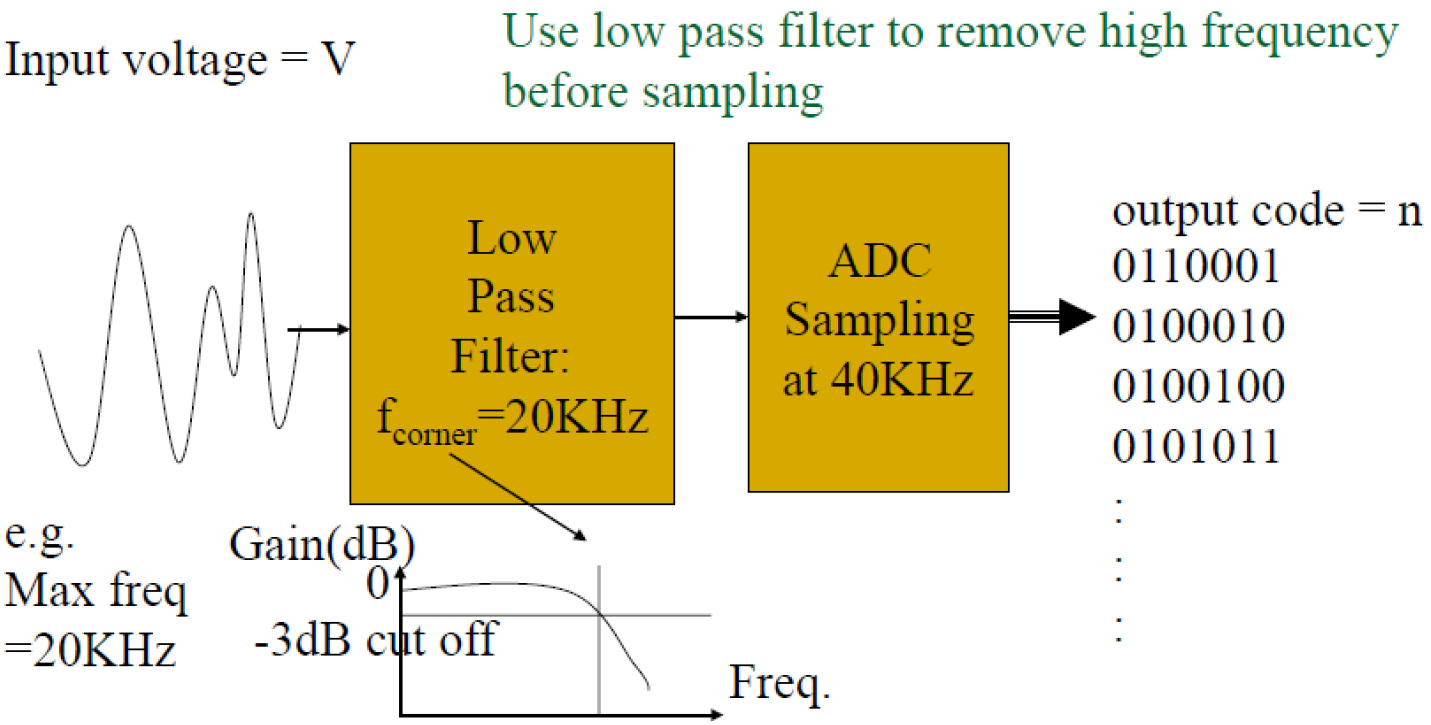
SAMPLING EXAMPLES (ALIASING)

Graphical Example 3:

- SR = 20,000 Hz
- Nyquist Frequency = 10,000 Hz
- f = 12,500 Hz, f' = 7,500



REDUCING EFFECTS OF ALIASING



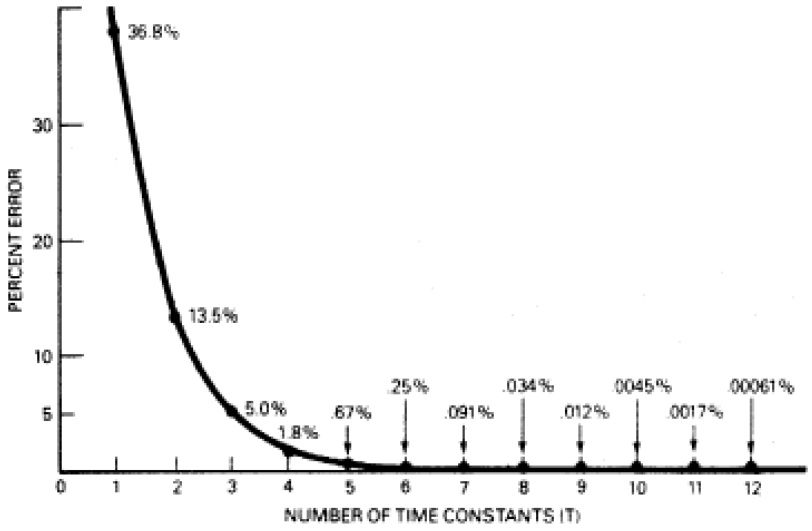
HARDWARE CONSTRAINTS

Hardware issues

- Sampling speed depends on bit resolution
  - Think time constants
  - $\text{Settling error} = e^{-\frac{t}{\tau}}$

Examples:

- 8-bit resolution takes 5.5τ
- 12-bit resolution takes 8.3τ
- 16-bit resolution takes 11τ



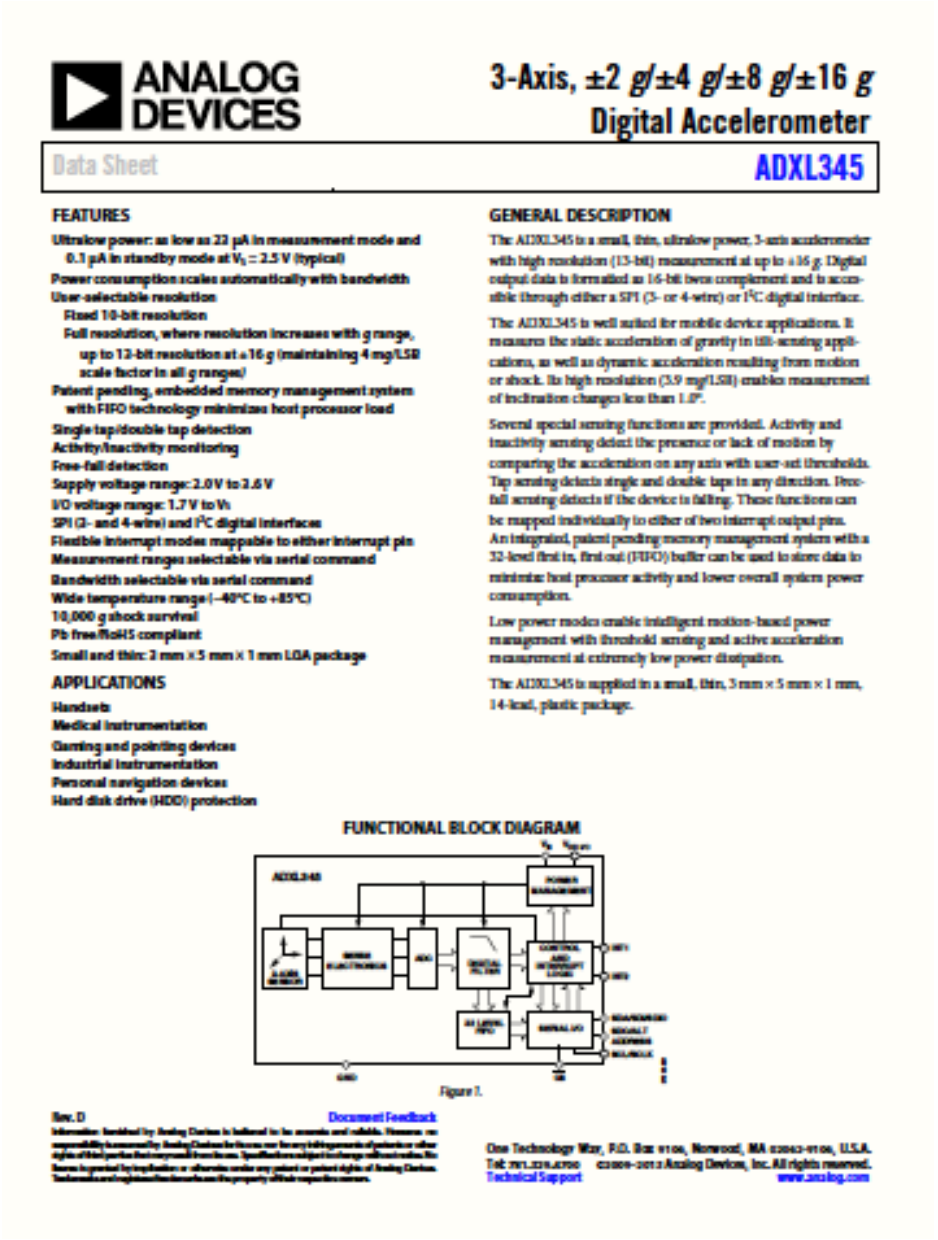
Output Settling Error as a Function of Number of Time Constants

Datel Data Acquisition and Conversion Handbook

READING A DATASHEET

Look for:

- Part number
- Basic specifications
- Functional block diagram or schematic diagram
- Date/revision number



ANALOG DEVICES

3-Axis, ±2 g/±4 g/±8 g/±16 g Digital Accelerometer

Data Sheet ADXL345

FEATURES

Ultralow power: as low as 22 µA in measurement mode and 0.1 µA in standby mode at V_S = 2.5 V (typical)
Power consumption scales automatically with bandwidth
User-selectable resolution
Fixed 10-bit resolution
Full resolution, where resolution increases with g range, up to 13-bit resolution at ±16 g (maintaining 4 mg/LSB scale factor in all g ranges)
Patent pending, embedded memory management system with FIFO technology minimizes host processor load
Single tap/double tap detection
Activity/inactivity monitoring
Free-fall detection
Supply voltage range: 2.0 V to 3.6 V
I/O voltage range: 1.7 V to V_S
SPI (3- and 4-wire) and I²C digital interfaces
Flexible interrupt modes mappable to either interrupt pin
Measurement ranges selectable via serial command
Bandwidth selectable via serial command
Wide temperature range (−40°C to +85°C)
10,000 g shock survival
Pb free/RoHS compliant
Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS

Handsets
Medical instrumentation
Gaming and pointing devices
Industrial instrumentation
Personal navigation devices
Hard disk drive (HDD) protection

GENERAL DESCRIPTION

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit two's complement and is accessible through either a SPI (3- or 4-wire) or I²C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated, patent pending memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

Figure 1.

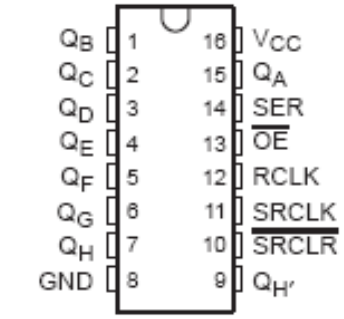
Rev. D Document Feedback

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 ©2009–2013 Analog Devices, Inc. All rights reserved.
Technical Support www.analog.com

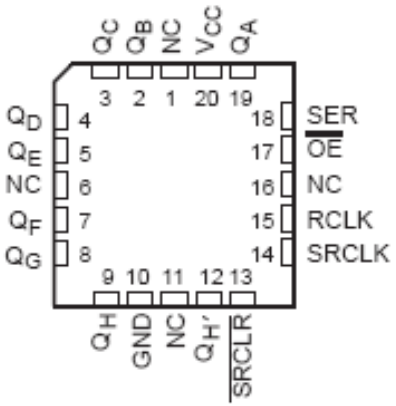
PINOUT OR CONNECTION DIAGRAM

Lists pins, their functions, and where they're physically located

SN54HC595 . . . J OR W PACKAGE
SN74HC595 . . . D, DB, DW, N, OR NS PACKAGE
(TOP VIEW)



SN54HC595 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

ABSOLUTE MAXIMUM RATINGS

Tells you what you can do without destroying the device, not the conditions under which it operates correctly.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration	
Any Axis, Unpowered	10,000 *g*
Any Axis, Powered	10,000 *g*
V_S	−0.3 V to +3.6 V
$V_{DD\ I/O}$	−0.3 V to +3.6 V
Digital Pins	−0.3 V to $V_{DD\ I/O}$ + 0.3 V or 3.6 V, whichever is less
All Other Pins	−0.3 V to +3.6 V
Output Short-Circuit Duration (Any Pin to Ground)	Indefinite
Temperature Range	
Powered	−40°C to +105°C
Storage	−40°C to +105°C

RECOMMENDED OPERATING CONDITIONS

Recommended settings for various parts of the device.

recommended operating conditions (see Note 3)

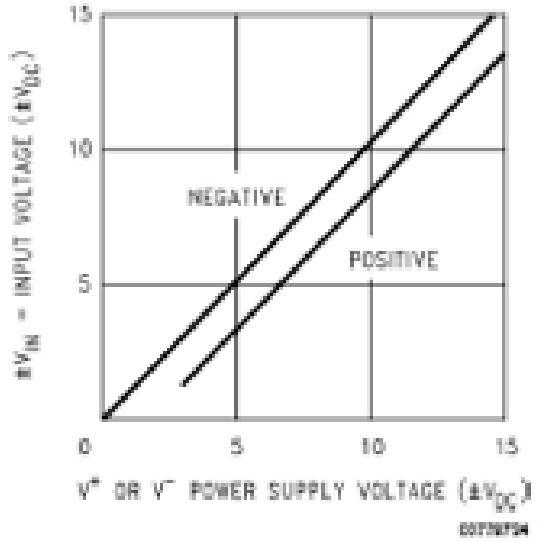
			SN54HC595			SN74HC595			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		2	5	6	2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			1.5			V
		V_{CC} = 4.5 V	3.15			3.15			
		V_{CC} = 6 V	4.2			4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V			0.5			0.5	V
		V_{CC} = 4.5 V			1.35			1.35	
		V_{CC} = 6 V			1.8			1.8	
V_I	Input voltage		0		V_{CC}	0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	0		V_{CC}	V
$\Delta t/\Delta v$‡	Input transition rise/fall time	V_{CC} = 2 V			1000			1000	ns
		V_{CC} = 4.5 V			500			500	
		V_{CC} = 6 V			400			400	
T_A	Operating free-air temperature		−55		125	−40		85	°C

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

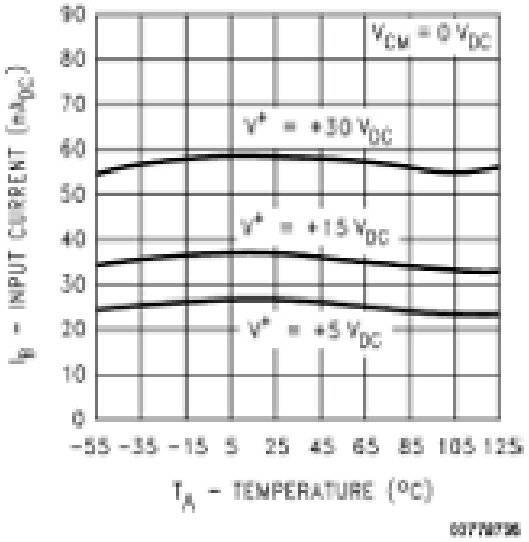
‡ If this device is used in the threshold region (from V_{IL}max = 0.5 V to V_{IH}min = 1.5 V), there is a potential to go into the wrong state from induced grounding, causing double clocking. Operating with the inputs at t_t = 1000 ns and V_{CC} = 2 V does not damage the device; however, functionally, the CLK inputs are not ensured while in the shift, count, or toggle operating modes.

Typical Performance Characteristics

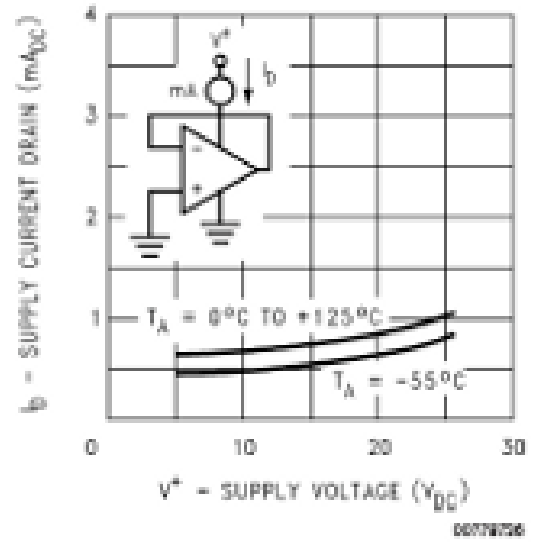
Input Voltage Range



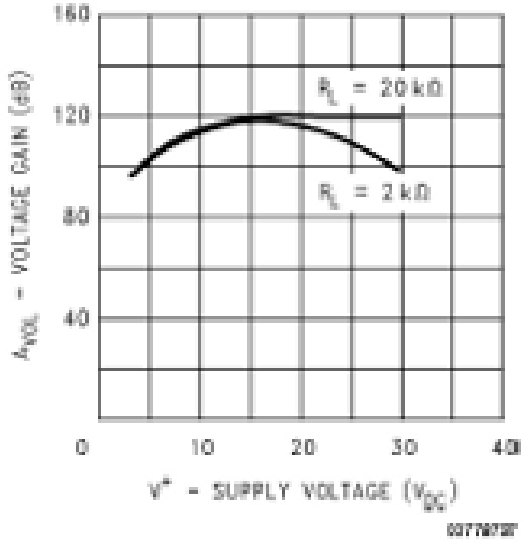
Input Current



Supply Current



Voltage Gain



OTHER THINGS

- Truth tables
- Timing diagrams
- Applications
- Schematics
- Physical dimensions