Máster Universitario en Software de Sistemas Distribuidos y Empotrados

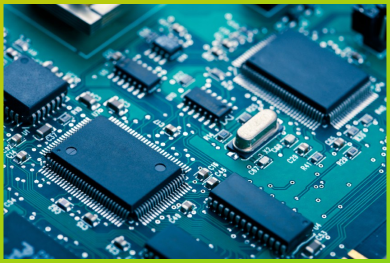
Universidad Politécnica de Madrid | ETSI SISTEMAS INFORMÁTICOS

Embedded Systems

Microcontrollers

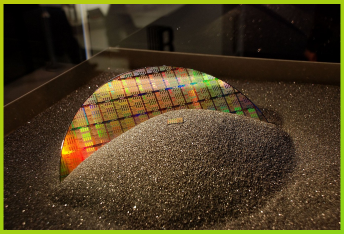
Targets

- To enable the use of µController **peripherals** based on the manufacturer's information.
- Training in the use of Embedded Systems (ES) **development environments**.
- To enable the use of common resources in some µControllers for **signal processing**.



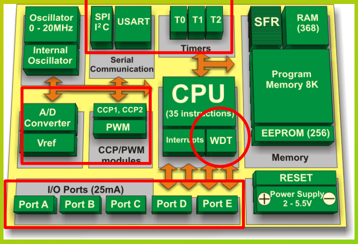
Major players in microchips fabrication

- Manufacturers of photolotographic systems.
  - ASML (accounts for the 67% of worldwide sales).
  - Ultratech.
  - Canon.
  - Nikon.
- Major integrated circuits manufacturers
  - INTEL (it is headquartered in EEUU and fabricates its own chips).
  - Samsung (it is headquartered in South Korea and fabricates for Qualcomm, IBM, Nvidia, Tesla, …).
  - TSMC (it is headquartered in Taiwan and fabricates for Apple, AMD, Qualcomm, Altera, Broadcom, Nvidia, Sony, Huawei, …).
  - GlobalFoundries (it is headquartered in EEUU and fabricates for AMD, Broadcom, Qualcomm, STMicroelectronics, …).



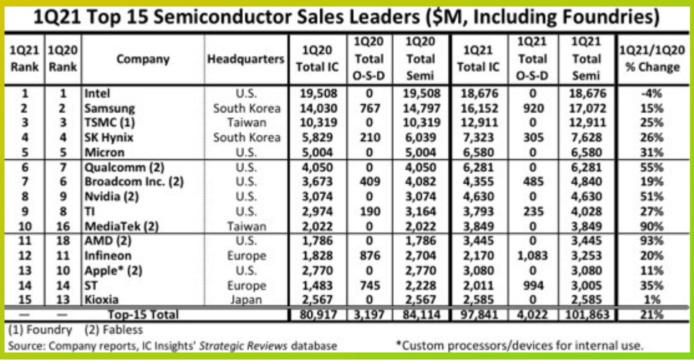
Microcontrollers

- They are "complete" computers (CPU, memory, oscillator + "previously" considered peripheral components) in a single integrated circuit.
- If peripheral components and memory are absent, we are talking about microprocessors.



Semiconductors market

(2021)



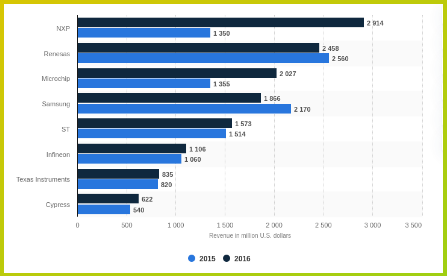
1Q21 Top 15 Semiconductor Sales Leaders ($M, Including Foundries)

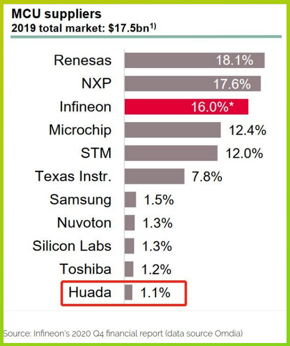
1Q21 Rank	1Q20 Rank	Company	Headquarters	1Q20 Total IC	1Q20 Total O-S-D	1Q20 Total Semi	1Q21 Total IC	1Q21 Total O-S-D	1Q21 Total Semi	1Q21/1Q20 % Change
1	1	Intel	U.S.	19,508	0	19,508	18,676	0	18,676	-4%
2	2	Samsung	South Korea	14,030	767	14,797	16,152	920	17,072	15%
3	3	TSMC (1)	Taiwan	10,319	0	10,319	12,911	0	12,911	25%
4	4	SK Hynix	South Korea	5,829	210	6,039	7,323	305	7,628	26%
5	5	Micron	U.S.	5,004	0	5,004	6,580	0	6,580	31%
6	7	Qualcomm (2)	U.S.	4,050	0	4,050	6,281	0	6,281	55%
7	6	Broadcom Inc. (2)	U.S.	3,673	409	4,082	4,355	485	4,840	19%
8	9	Nvidia (2)	U.S.	3,074	0	3,074	4,630	0	4,630	51%
9	8	TI	U.S.	2,974	190	3,164	3,793	235	4,028	27%
10	16	MediaTek (2)	Taiwan	2,022	0	2,022	3,849	0	3,849	90%
11	18	AMD (2)	U.S.	1,786	0	1,786	3,445	0	3,445	93%
12	11	Infineon	Europe	1,828	876	2,704	2,170	1,083	3,253	20%
13	10	Apple* (2)	U.S.	2,770	0	2,770	3,080	0	3,080	11%
14	14	ST	Europe	1,483	745	2,228	2,011	994	3,005	35%
15	13	Kioxia	Japan	2,567	0	2,567	2,585	0	2,585	1%
—	—	Top-15 Total		80,917	3,197	84,114	97,841	4,022	101,863	21%

(1) Foundry (2) Fabless

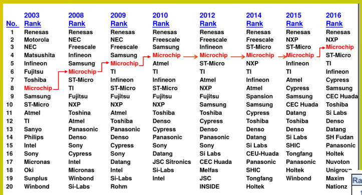
Source: Company reports, IC Insights' *Strategic Reviews* database *Custom processors/devices for internal use.

Microcontrollers market

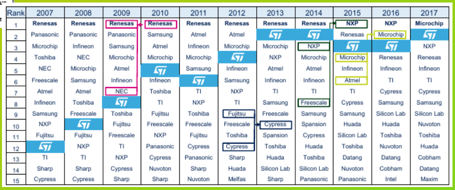




Microcontrollers market

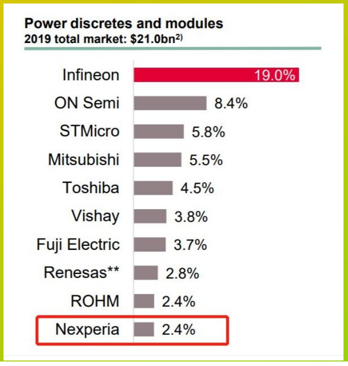


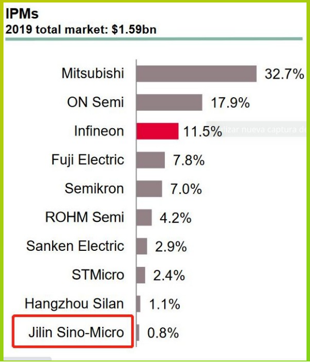
No.	2003 Rank	2008 Rank	2009 Rank	2010 Rank	2012 Rank	2014 Rank	2015 Rank	2016 Rank
1	Renesas	Renesas	Renesas	Renesas	Renesas	Renesas	Renesas	Renesas
2	Motorola	NEC	NEC	Freescale	Freescale	Freescale	NXP	NXP
3	NEC	Freescale	Freescale	Samsung	Infineon	ST-Micro	ST-Micro	Microchip
4	Matsushita	Infineon	Samsung	Microchip	Microchip	Microchip	Microchip	ST-Micro
5	Infineon	Samsung	Microchip	Atmel	ST-Micro	NXP	Infineon	TI
6	Fujitsu	Microchip	TI	TI	TI	TI	TI	Infineon
7	Toshiba	ST-Micro	Infineon	Infineon	Atmel	Infineon	Atmel	Cypress
8	Microchip	TI	ST-Micro	ST-Micro	NXP	Atmel	Cypress	Samsung
9	Samsung	Fujitsu	Fujitsu	Fujitsu	Fujitsu	Spansion	Samsung	CEC Huada
10	ST-Micro	NXP	NXP	NXP	Samsung	Samsung	CEC Huada	Toshiba
11	Atmel	Toshina	Atmel	Toshiba	Toshiba	Cypress	Datang	Si Labs
12	TI	Atmel	Toshiba	Denso	Cypress	Toshiba	Toshiba	Denso
13	Sanyo	Panasonic	Panasonic	Cypress	Denso	Denso	Denso	Datang
14	Philips	Denso	Denso	Panasonic	Panasonic	Datang	Si Labs	SH Fudan
15	Intel	Sony	Cypress	Sony	Sony	Si Labs	SHIC	Panasonic
16	Sony	Cypress	Sony	Datang	Si Labs	CEU-Huada	Tongfang	Holtek
17	Micronas	Intel	Datang	JSC Sitronics	CEC Huada	Panasonic	Panasonic	Nuvoton
18	Oki	Micronas	Intel	Si-Labs	Melfas	SHIC	Holtek	Unigrou
19	Sunplus	Winbond	Si-Labs	Intel	JSC	Tongfang	Winbond	Maxim
20	Winbond	Si-Labs	Rohm		INSIDE	Holtek		Nationz



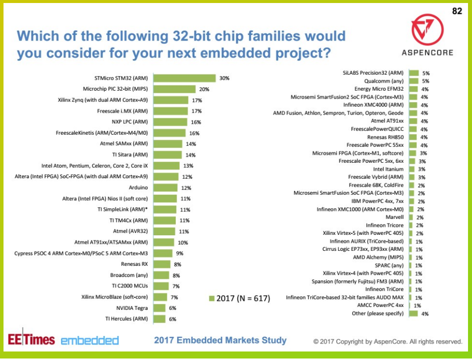
Rank	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
1	Renesas	Renesas	Renesas	Renesas	Renesas	Renesas	Renesas	Renesas	NXP	NXP	Microchip
2	Panasonic	Panasonic	Panasonic	Samsung	Atmel	Atmel	ST	ST	Renesas	Microchip	ST
3	Microchip	Infineon	Samsung	Atmel	Infineon	Microchip	Microchip	NXP	ST	ST	NXP
4	Toshiba	NEC	Microchip	Microchip	Microchip	ST	NXP	Microchip	Microchip	Renesas	Renesas
5	NEC	Microchip	Atmel	ST	Samsung	Infineon	Atmel	Atmel	Infineon	Infineon	Infineon
6	Freescale	Samsung	Infineon	Infineon	ST	Samsung	Infineon	Infineon	Atmel	TI	TI
7	Atmel	Atmel	NEC	Toshiba	TI	NXP	TI	TI	TI	Cypress	Cypress
8	Infineon	Toshiba	ST	TI	NXP	TI	Samsung	Freescale	Cypress	Samsung	Samsung
9	Samsung	Freescale	Toshiba	Fujitsu	Toshiba	Fujitsu	Freescale	Samsung	Samsung	Silicon Lab	Silicon Lab
10	NXP	ST	Fujitsu	Freescale	Freescale	Freescale	Cypress	Spansion	Huada	Huada	Nuvoton
11	Fujitsu	Fujitsu	Freescale	NXP	Fujitsu	Toshiba	Spansion	Cypress	Silicon Lab	Toshiba	Toshiba
12	ST	NXP	TI	Panasonic	Cypress	Cypress	Toshiba	Huada	Toshiba	Nuvoton	Huada
13	TI	TI	NXP	Cypress	Panasonic	Sharp	Huada	Toshiba	Datang	Datang	Cobham
14	Sharp	Sharp	Cypress	Nuvoton	Sharp	Huada	Silicon Lab	Silicon Lab	Nuvoton	Cobham	Datang
15	Cypress	Cypress	Sharp	Sharp	Nuvoton	Melfas	Sharp	Panasonic	Panasonic	Intel	Maxim

Power devices market

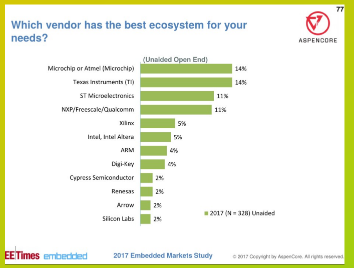


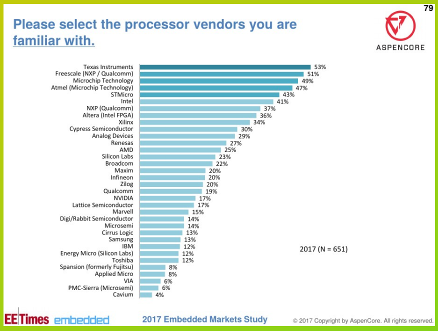


Perception of the microcontroller market

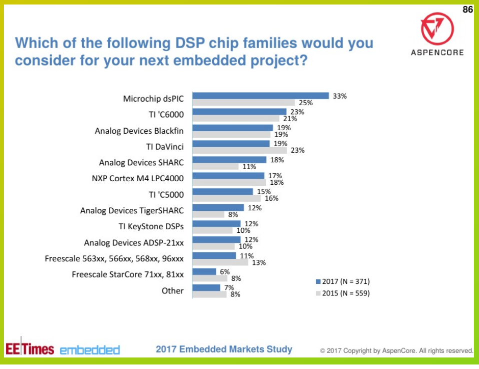


Perception of the microcontroller market



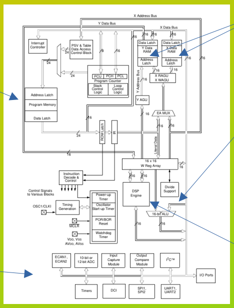


Perception of the microcontroller market

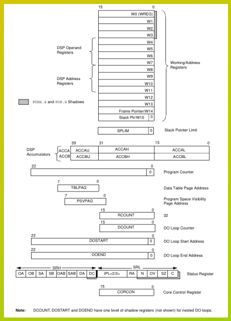


Practical example. dsPIC33F

Block diagram

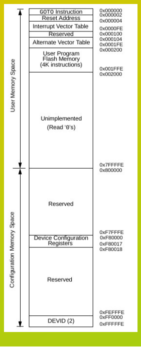


dsPIC33F
Program model



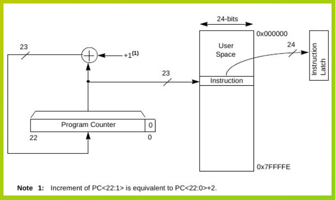
Register(s) Name	Description
W0 through W15	Working register array
ACCA, ACCB	40-bit DSP Accumulators
PC	23-bit Program Counter
SR	ALU and DSP Engine Status register
SPLIM	Stack Pointer Limit Value register
TBLPAG	Table Memory Page Address register
PSVPAG	Program Space Visibility Page Address register
RCOUNT	REPEAT Loop Count register
DCOUNT	DO Loop Count register
DOSTART	DO Loop Start Address register
DOEND	DO Loop End Address register
CORCON	Contains DSP Engine and DO Loop control bits

dsPIC33F
Program memory

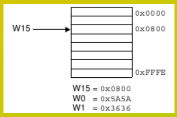


Program memory generic map

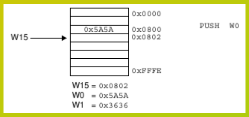
Fetch



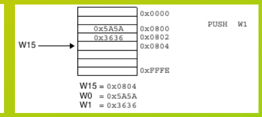
dsPIC33F
Memory and stack pointer operation



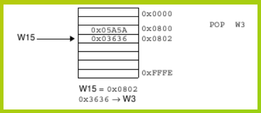
Device reset



Push

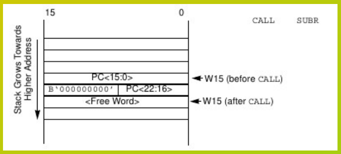


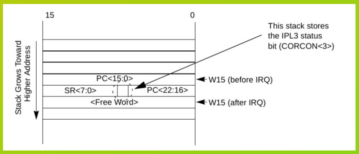
Push



Pop

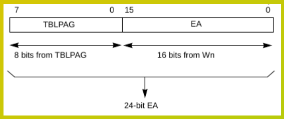
dsPIC33F
Memory and stack pointer operation

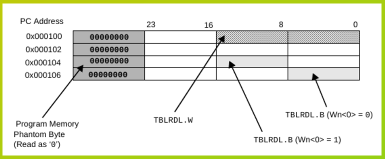


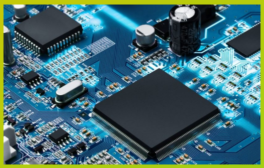


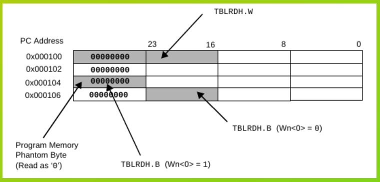
15
0
This stack stores the IPL3 status bit (CORCON<3>)
Stack Grows Toward Higher Address
PC<15:0>
SR<7:0>
PC<22:16>
<Free Word>
W15 (before IRQ)
W15 (after IRQ)

dsPIC33F
Program memory access using table instructions

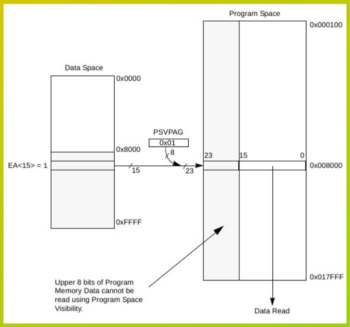


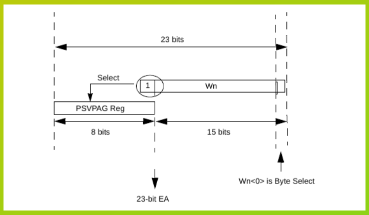






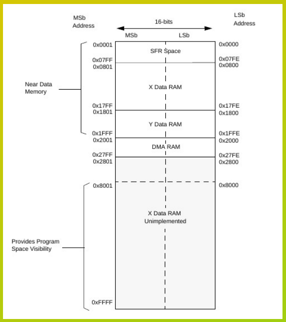
dsPIC33F
Program memory visibility using from data space



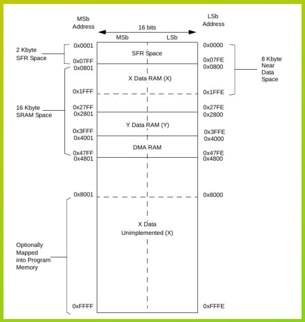


dsPIC33F

Data memory



Generic data memory for dsPIC33F Family

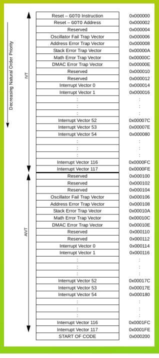


Data memory of dsPIC33FJ128MC802

dsPIC33F

Interrupts

- The CPU can operate at 16 priority levels (0..15).
- The current CPU priority is encoded in:
  - IPL<0:2> in SR register.
  - IPL3 in CORCON register.
- For an interrupt to be attended its priority must be greater than current CPU priority.
- Traps (level 8 and above) cannot be masked.
- The IPCx registers are used to assign the priority of an interrupt.
- The IFSx registers allow the flag of an interrupt to be updated.
- The IECx registers allow interrupts to be enabled.



dsPIC33F

Interrupts

- The XC16 compiler
  - Provides meaningful names for each interrupt routine (/.../microchip/xc16/vX.XX/docs/vector_docs)
  - As a prologue to the interrupt processing, it saves the work registers (W0..W15), the status register (SR) and the counter for the Repeat instruction. If you want to save something else, you can specify it as a parameter in the interrupt processing routine.

```
void __attribute__((interrupt(no_auto_psv, save(var1, var2)))) isr0(void);
```

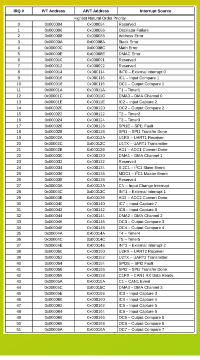
  - If you are interested in the option of quick context saving (PUSH.S and POP.S)

```
void __attribute__((interrupt(no_auto_psv, shadow))) isr0(void);
```

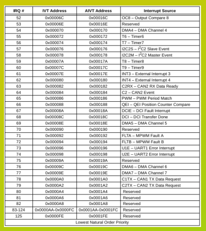
  - There are more options (see compiler manual).

dsPIC33F

Interrupts



IRQ #	IVT Address	AIVT Address	Interrupt Source
Highest Natural Order Priority			
0	0x000004	0x000084	Reserved
1	0x000006	0x000086	Oscillator Failure
2	0x000008	0x000088	Address Error
3	0x00000A	0x00008A	Stack Error
4	0x00000C	0x00008C	Math Error
5	0x00000E	0x00008E	DMAC Error
6	0x000010	0x000091	Reserved
7	0x000012	0x000092	Reserved
8	0x000014	0x000114	INT0 – External Interrupt 0
9	0x000016	0x000116	IC1 – Input Compare 1
10	0x000018	0x000118	OC1 – Output Compare 1
11	0x00001A	0x00011A	T1 – Timer1
12	0x00001C	0x00011C	DMA0 – DMA Channel 0
13	0x00001E	0x00011E	IC2 – Input Capture 2
14	0x000020	0x000120	OC2 – Output Compare 2
15	0x000022	0x000122	T2 – Timer2
16	0x000024	0x000124	T3 – Timer3
17	0x000026	0x000126	SPI1E – SPI1 Fault
18	0x000028	0x000128	SPI1 – SPI1 Transfer Done
19	0x00002A	0x00012A	U1RX – UART1 Receiver
20	0x00002C	0x00012C	U1TX – UART1 Transmitter
21	0x00002E	0x00012E	AD1 – ADC1 Convert Done
22	0x000030	0x000130	DMA1 – DMA Channel 1
23	0x000032	0x000132	Reserved
24	0x000034	0x000134	SI2C1 – I^{2}C1 Slave Event
25	0x000036	0x000136	MI2C1 – I^{2}C1 Master Event
26	0x000038	0x000138	Reserved
27	0x00003A	0x00013A	CN – Input Change Interrupt
28	0x00003C	0x00013C	INT1 – External Interrupt 1
29	0x00003E	0x00013E	AD2 – ADC2 Convert Done
30	0x000040	0x000140	IC7 – Input Capture 7
31	0x000042	0x000142	IC8 – Input Capture 8
32	0x000044	0x000144	DMA2 – DMA Channel 2
33	0x000046	0x000146	OC3 – Output Compare 3
34	0x000048	0x000148	OC4 – Output Compare 4
35	0x00004A	0x00014A	T4 – Timer4
36	0x00004C	0x00014C	T5 – Timer5
37	0x00004E	0x00014E	INT2 – External Interrupt 2
38	0x000050	0x000150	U2RX – UART2 Receiver
39	0x000052	0x000152	U2TX – UART2 Transmitter
40	0x000054	0x000154	SPI2E – SPI2 Fault
41	0x000056	0x000156	SPI2 – SPI2 Transfer Done
42	0x000058	0x000158	C1RX – CAN1 RX Data Ready
43	0x00005A	0x00015A	C1 – CAN1 Event
44	0x00005C	0x00015C	DMA3 – DMA Channel 3
45	0x00005E	0x00015E	IC3 – Input Capture 3
46	0x000060	0x000160	IC4 – Input Capture 4
47	0x000062	0x000162	IC5 – Input Capture 5
48	0x000064	0x000164	IC6 – Input Capture 6
49	0x000066	0x000166	OC5 – Output Compare 5
50	0x000068	0x000168	OC6 – Output Compare 6
51	0x00006A	0x00016A	OC7 – Output Compare 7



IRQ #	IVT Address	AIVT Address	Interrupt Source
52	0x00006C	0x00016C	OC8 – Output Compare 8
53	0x00006E	0x00016E	Reserved
54	0x000070	0x000170	DMA4 – DMA Channel 4
55	0x000072	0x000172	T6 – Timer6
56	0x000074	0x000174	T7 – Timer7
57	0x000076	0x000176	I2C2S – I^{2}C2 Slave Event
58	0x000078	0x000178	I2C2M – I^{2}C2 Master Event
59	0x00007A	0x00017A	T8 – Timer8
60	0x00007C	0x00017C	T9 – Timer9
61	0x00007E	0x00017E	INT3 – External Interrupt 3
62	0x000080	0x000180	INT4 – External Interrupt 4
63	0x000082	0x000182	C2RX – CAN2 RX Data Ready
64	0x000084	0x000184	C2 – CAN2 Event
65	0x000086	0x000186	PWM – PWM Period Match
66	0x000088	0x000188	QEI – QEI Position Counter Compare
67	0x00008A	0x00018A	DCIE – DCI Fault Interrupt
68	0x00008C	0x00018C	DCI – DCI Transfer Done
69	0x00008E	0x00018E	DMA5 – DMA Channel 5
70	0x000090	0x000190	Reserved
71	0x000092	0x000192	FLTA – MPWM Fault A
72	0x000094	0x000194	FLTB – MPWM Fault B
73	0x000096	0x000196	U1E – UART1 Error Interrupt
74	0x000098	0x000198	U2E – UART2 Error Interrupt
75	0x00009A	0x00019A	Reserved
76	0x00009C	0x00019C	DMA6 – DMA Channel 6
77	0x00009E	0x00019E	DMA7 – DMA Channel 7
78	0x0000A0	0x0001A0	C1TX – CAN1 TX Data Request
79	0x0000A2	0x0001A2	C2TX – CAN2 TX Data Request
80	0x0000A4	0x0001A4	Reserved
81	0x0000A6	0x0001A6	Reserved
82	0x0000A8	0x0001A8	Reserved
83-124	0x0000AA-0x0000FC	0x0001AA-0x0001FC	Reserved
125	0x0000FE	0x0001FE	Reserved
Lowest Natural Order Priority			

- Interruptions with the same priority level resolve the conflict by applying "natural order".
- The dsPIC33F has four possible sources of non-maskable interrupts:
  - Oscillator failure.
  - Address error.
  - Stack error.
  - Math error.
  - DMAC error.

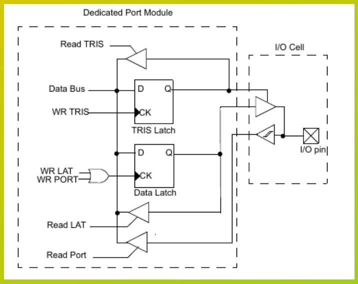
dsPIC33F

Some outstanding peripherals and configuration features

- Input output ports.
- System clock signal generator
- Timers.
- USART.
- CAD.

dsPIC33F

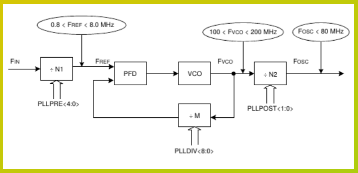
IO Ports



- TRIS register. It specifies the direction in which digital data travels.
- PORT register. It is used to read or write to a port.
- LAT register. It is used to write to a port or to read the last write to the output latch.

dsPIC33F

System clock signal generator



$$F_{OSC} = F_{IN} \times \left(\frac{M}{N1 \times N2}\right) = F_{IN} \times \left(\frac{(PLLDIV + 2)}{(PLLPRE + 2) \times 2(PLLPOST + 1)}\right)$$

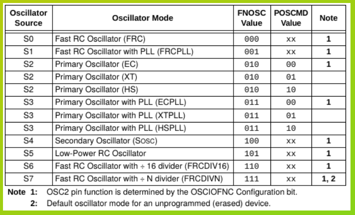
Where,

$N1$ = PLLPRE + 2

$N2$ = 2 x (PLLPOST + 1)

M = PLLDIV + 2

- In several of the possible oscillator modes we can use a frequency multiplier (PLL)



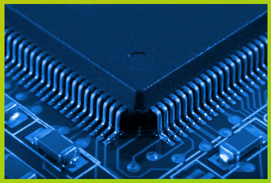
Oscillator Source	Oscillator Mode	FNOSC Value	POSCMD Value	Note
S0	Fast RC Oscillator (FRC)	000	xx	**1**
S1	Fast RC Oscillator with PLL (FRCPLL)	001	xx	**1**
S2	Primary Oscillator (EC)	010	00	**1**
S2	Primary Oscillator (XT)	010	01	
S2	Primary Oscillator (HS)	010	10	
S3	Primary Oscillator with PLL (ECPLL)	011	00	**1**
S3	Primary Oscillator with PLL (XTPLL)	011	01	
S3	Primary Oscillator with PLL (HSPLL)	011	10	
S4	Secondary Oscillator (SOSC)	100	xx	**1**
S5	Low-Power RC Oscillator	101	xx	**1**
S6	Fast RC Oscillator with ÷ 16 divider (FRCDIV16)	110	xx	**1**
S7	Fast RC Oscillator with ÷ N divider (FRCDIVN)	111	xx	**1, 2**

Note 1: OSC2 pin function is determined by the OSCIOFNC Configuration bit.
2: Default oscillator mode for an unprogrammed (erased) device.

dsPIC33F

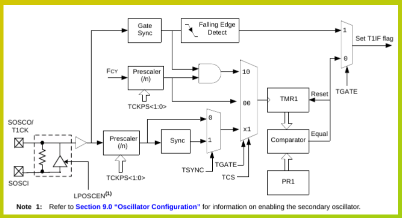
System clock signal generator

```
void init_micro(void)
{
   // Configure Oscillator to operate the device at ?? Mhz
   // Fosc = Fin*M/(N1*N2), Fcy = Fosc/2
   // Fosc = ??
   // Fcy = ??

   // Configure PLL prescaler, PLL postscaler and PLL divisor

   PLLFBDbits.PLLDIV = xx; // M = PLLDIV + 2 = xx + 2 → PLLDIV = M - 2
   CLKDIVbits.PLLPOST = yy; // N2 = 2*(PLLPOST + 1) → PLLPOST = (N2 / 2) - 1
   CLKDIVbits.PLLPRE = zz; // N1 = PLLPRE + 2 → PLLPRE = N1 -2

   // clock switching to incorporate PLL
   __builtin_write_OSCCONH(0x03); // Initiate Clock Switch to Primary
   __builtin_write_OSCCONL(0x01); // Start clock switching

   while (OSCCONbits.COSC != 0b011); // Wait for Clock switch to occur
   while (OSCCONbits.LOCK != 1) {}; // Wait for PLL to lock (If LOCK = 1 -> PLL start-up timer is satisfied)
}
```

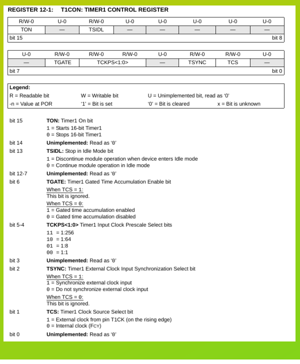


dsPIC33F

Timers



Note 1: Refer to Section 9.0 "Oscillator Configuration" for information on enabling the secondary oscillator.



REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7							bit 0

Legend:
R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 15	**TON:** Timer1 On bit 1 = Starts 16-bit Timer1 0 = Stops 16-bit Timer1
bit 14	**Unimplemented:** Read as '0'
bit 13	**TSIDL:** Stop in Idle Mode bit 1 = Discontinue module operation when device enters Idle mode 0 = Continue module operation in Idle mode
bit 12-7	**Unimplemented:** Read as '0'
bit 6	**TGATE:** Timer1 Gated Time Accumulation Enable bit When TCS = 1: This bit is ignored. When TCS = 0: 1 = Gated time accumulation enabled 0 = Gated time accumulation disabled
bit 5-4	**TCKPS<1:0>** Timer1 Input Clock Prescale Select bits 11 = 1:256 10 = 1:64 01 = 1:8 00 = 1:1
bit 3	**Unimplemented:** Read as '0'
bit 2	**TSYNC:** Timer1 External Clock Input Synchronization Select bit When TCS = 1: 1 = Synchronize external clock input 0 = Do not synchronize external clock input When TCS = 0: This bit is ignored.
bit 1	**TCS:** Timer1 Clock Source Select bit 1 = External clock from pin T1CK (on the rising edge) 0 = Internal clock (FCY)
bit 0	**Unimplemented:** Read as '0'

dsPIC33F

Timers

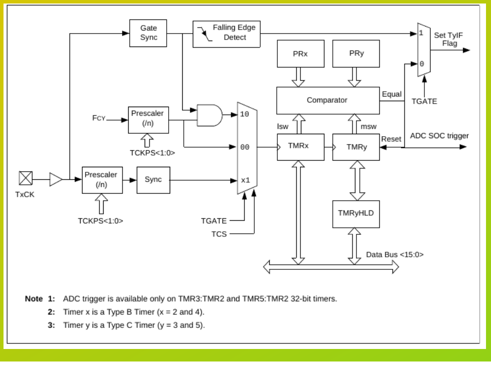
- It is possible to work with 32-bit timers from two 16-bit timers.

TYPE B timer (lsw)	TYPE C timer (msw)
Timer2	Timer3
Timer4	Timer5
Timer6	Timer7
Timer8	Timer9

- The 32-bit timers are configured using the type B timer registers (with the exception of the TSIDL bit).
- The 32-bit timers use the interrupt processing, priority, interrupt flag, and enable flag of the type C timer.

dsPIC33F

Timers

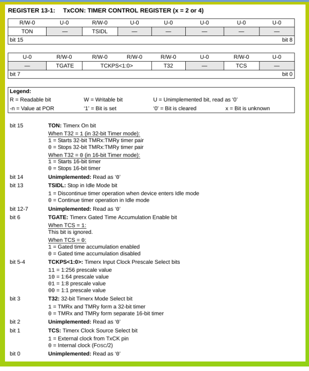


Note 1: ADC trigger is available only on TMR3:TMR2 and TMR5:TMR2 32-bit timers.
2: Timer x is a Type B Timer (x = 2 and 4).
3: Timer y is a Type C Timer (y = 3 and 5).

- TIMERx tipo B
- TIMERy tipo C

dsPIC33F

Timers



REGISTER 13-1: TxCON: TIMER CONTROL REGISTER (x = 2 or 4)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

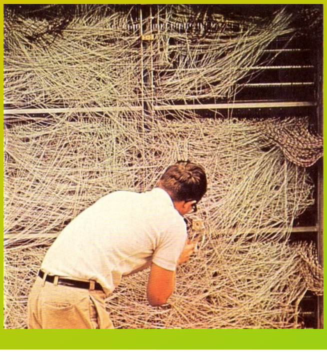
U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		T32	—	TCS	—
bit 7							bit 0

Legend:
R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 15	**TON:** Timerx On bit When T32 = 1 (in 32-bit Timer mode): 1 = Starts 32-bit TMRx:TMRy timer pair 0 = Stops 32-bit TMRx:TMRy timer pair When T32 = 0 (in 16-bit Timer mode): 1 = Starts 16-bit timer 0 = Stops 16-bit timer
bit 14	**Unimplemented:** Read as '0'
bit 13	**TSIDL:** Stop in Idle Mode bit 1 = Discontinue timer operation when device enters Idle mode 0 = Continue timer operation in Idle mode
bit 12-7	**Unimplemented:** Read as '0'
bit 6	**TGATE:** Timerx Gated Time Accumulation Enable bit When TCS = 1: This bit is ignored. When TCS = 0: 1 = Gated time accumulation enabled 0 = Gated time accumulation disabled
bit 5-4	**TCKPS<1:0>:** Timerx Input Clock Prescale Select bits 11 = 1:256 prescale value 10 = 1:64 prescale value 01 = 1:8 prescale value 00 = 1:1 prescale value
bit 3	**T32:** 32-bit Timerx Mode Select bit 1 = TMRx and TMRy form a 32-bit timer 0 = TMRx and TMRy form separate 16-bit timer
bit 2	**Unimplemented:** Read as '0'
bit 1	**TCS:** Timerx Clock Source Select bit 1 = External clock from TxCK pin 0 = Internal clock (FOSC/2)
bit 0	**Unimplemented:** Read as '0'

dsPIC33F

USART



- It allows serial communication between two (or more) devices.
- It can work with only three wires: transmission, reception and ground.

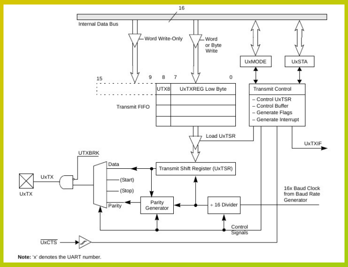
dsPIC33F

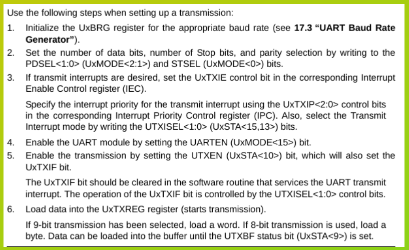
USART

- Registers
  - UxMODE. It allows to stablish mode of operation.
  - UxSTAT. It allows to read the status of the device.
  - UxRXREG. Receive register. Stores the last data received.
  - UxTXREG. Transmite register. Allows to send data.
  - UxBRG. Baud rate register. It is used to set the speed of communications.

dsPIC33F

USART. Transmitter block diagram





Use the following steps when setting up a transmission:

1. Initialize the UxBRG register for the appropriate baud rate (see **17.3 "UART Baud Rate Generator"**).
2. Set the number of data bits, number of Stop bits, and parity selection by writing to the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.
3. If transmit interrupts are desired, set the UxTXIE control bit in the corresponding Interrupt Enable Control register (IEC).

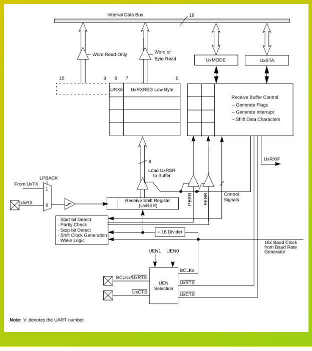
   Specify the interrupt priority for the transmit interrupt using the UxTXIP<2:0> control bits in the corresponding Interrupt Priority Control register (IPC). Also, select the Transmit Interrupt mode by writing the UTXISEL<1:0> (UxSTA<15,13>) bits.
4. Enable the UART module by setting the UARTEN (UxMODE<15>) bit.
5. Enable the transmission by setting the UTXEN (UxSTA<10>) bit, which will also set the UxTXIF bit.

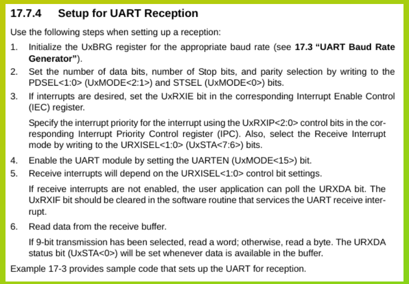
   The UxTXIF bit should be cleared in the software routine that services the UART transmit interrupt. The operation of the UxTXIF bit is controlled by the UTXISEL<1:0> control bits.
6. Load data into the UxTXREG register (starts transmission).

   If 9-bit transmission has been selected, load a word. If 8-bit transmission is used, load a byte. Data can be loaded into the buffer until the UTXBF status bit (UxSTA<9>) is set.

dsPIC33F

USART. Receiver block diagram





17.7.4 Setup for UART Reception

Use the following steps when setting up a reception:

1. Initialize the UxBRG register for the appropriate baud rate (see **17.3 "UART Baud Rate Generator"**).
2. Set the number of data bits, number of Stop bits, and parity selection by writing to the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.
3. If interrupts are desired, set the UxRXIE bit in the corresponding Interrupt Enable Control (IEC) register.

   Specify the interrupt priority for the interrupt using the UxRXIP<2:0> control bits in the corresponding Interrupt Priority Control register (IPC). Also, select the Receive Interrupt mode by writing to the URXISEL<1:0> (UxSTA<7:6>) bits.
4. Enable the UART module by setting the UARTEN (UxMODE<15>) bit.
5. Receive interrupts will depend on the URXISEL<1:0> control bit settings.

   If receive interrupts are not enabled, the user application can poll the URXDA bit. The UxRXIF bit should be cleared in the software routine that services the UART receive interrupt.
6. Read data from the receive buffer.

   If 9-bit transmission has been selected, read a word; otherwise, read a byte. The URXDA status bit (UxSTA<0>) will be set whenever data is available in the buffer.

Example 17-3 provides sample code that sets up the UART for reception.

dsPIC33F

USART. Baud rate

Equation 17-1: UART Baud Rate (BRGH = 0)

$$Baud\ Rate = \frac{F_{CY}}{16 \times (UxBRG + 1)} \quad ......(1)$$

$$UxBRG = \frac{F_{CY}}{16 \times Baud\ Rate} - 1 \quad ......(2)$$

Note: F_{CY} denotes the instruction cycle clock frequency (FOSC/2).

Equation 17-2: UART Baud Rate (BRGH = 1)

$$Baud\ Rate = \frac{F_{CY}}{4 \times (UxBRG + 1)} \quad ......(1)$$

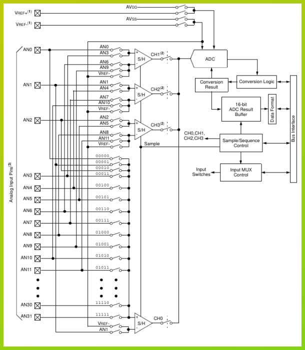
$$UxBRG = \frac{F_{CY}}{4 \times Baud\ Rate} - 1 \quad ......(2)$$

Note: F_{CY} denotes the instruction cycle clock frequency.

dsPIC33F

CAD

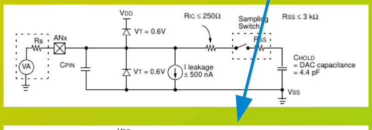
- Allows continuous and automatic sampling and conversion of single or multiple channels (up to 1.1 Msps).
- The dsPIC33F family has devices with 32 analog channels.
- External pins are available to set reference voltages (this allows to increase the resolution of the measurement).
- A 10-bit or 12-bit conversion is selectable.



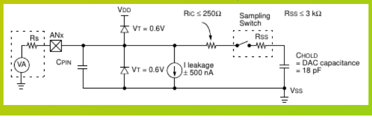
dsPIC33F
CAD

Total time of conversion

Sample interval | Conversion interval



10 bits



12 bits

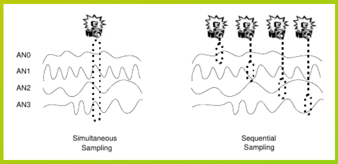
$$T_{AD} = T_{CY}(ADCS + 1)$$

$$ADCS = \frac{T_{AD}}{T_{CY}} - 1$$

Minimum T_{AD} = 75 ns
A 10 bits conversion requires 12 T_{AD}
A 12 bits conversion requires 14 T_{AD}

dsPIC33F
CAD

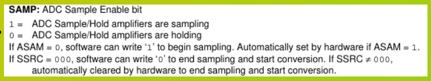
- In 10 bit mode:
  - Four independent S&H circuits could be used (CH0 to CH3).
  - Then multiple samples (simultaneous or sequential) and conversion are allowed in each ADC cycle.
- In 12 bit mode only CH0 S&H is used (and no simultaneous samples are allowed).

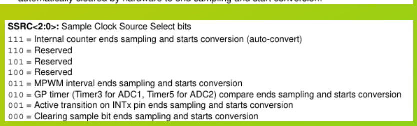


dsPIC33F
CAD

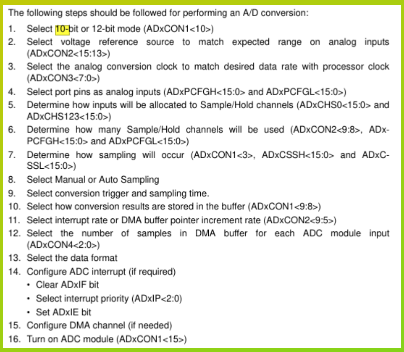
- Sampling and conversion could be started manually.
- Sample and conversion could be started automatically.
- It is also possible continuous data conversion (free running).

ADxCON1


SAMP: ADC Sample Enable bit
1 = ADC Sample/Hold amplifiers are sampling
0 = ADC Sample/Hold amplifiers are holding
If ASAM = 0, software can write '1' to begin sampling. Automatically set by hardware if ASAM = 1.
If SSRC = 000, software can write '0' to end sampling and start conversion. If SSRC ≠ 000, automatically cleared by hardware to end sampling and start conversion.


SSRC<2:0>: Sample Clock Source Select bits
111 = Internal counter ends sampling and starts conversion (auto-convert)
110 = Reserved
101 = Reserved
100 = Reserved
011 = MPWM interval ends sampling and starts conversion
010 = GP timer (Timer3 for ADC1, Timer5 for ADC2) compare ends sampling and starts conversion
001 = Active transition on INTx pin ends sampling and starts conversion
000 = Clearing sample bit ends sampling and starts conversion

dsPIC33F
CAD


The following steps should be followed for performing an A/D conversion:

1. Select 10-bit or 12-bit mode (ADxCON1<10>)
2. Select voltage reference source to match expected range on analog inputs (ADxCON2<15:13>)
3. Select the analog conversion clock to match desired data rate with processor clock (ADxCON3<7:0>)
4. Select port pins as analog inputs (ADxPCFGH<15:0> and ADxPCFGL<15:0>)
5. Determine how inputs will be allocated to Sample/Hold channels (ADxCHS0<15:0> and ADxCHS123<15:0>)
6. Determine how many Sample/Hold channels will be used (ADxCON2<9:8>, ADx-PCFGH<15:0> and ADxPCFGL<15:0>)
7. Determine how sampling will occur (ADxCON1<3>, ADxCSSH<15:0> and ADxC-SSL<15:0>)
8. Select Manual or Auto Sampling
9. Select conversion trigger and sampling time.
10. Select how conversion results are stored in the buffer (ADxCON1<9:8>)
11. Select interrupt rate or DMA buffer pointer increment rate (ADxCON2<9:5>)
12. Select the number of samples in DMA buffer for each ADC module input (ADxCON4<2:0>)
13. Select the data format
14. Configure ADC interrupt (if required)
    - Clear ADxIF bit
    - Select interrupt priority (ADxIP<2:0)
    - Set ADxIE bit
15. Configure DMA channel (if needed)
16. Turn on ADC module (ADxCON1<15>)

Thank you