
F28379D GPIO

HCMUTE

Monna Dang (Dang Hoang Anh Chuong)

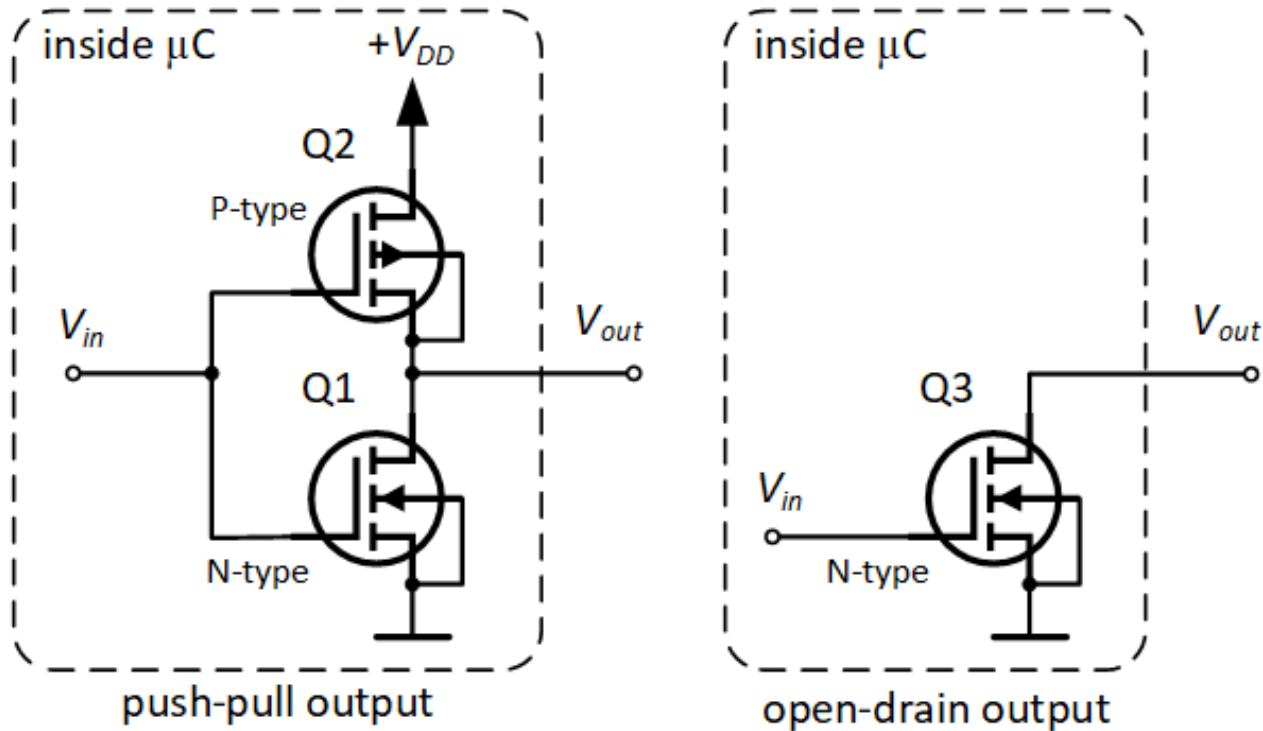
02/2024

Outline

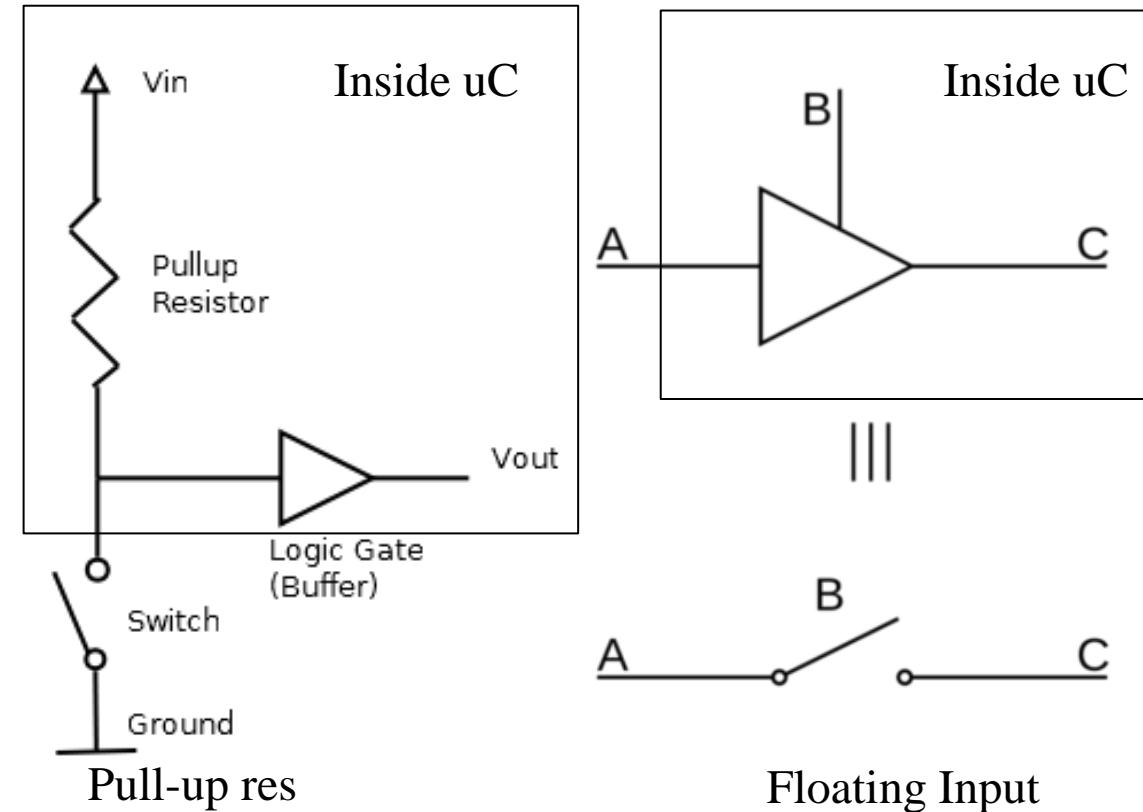
- 1. Overview**
- 2. GPIO**
- 3. Input Qualification**
- 4. Init a project**
- 5. Register**
- 6. Example**

1. Overview

Typical configurations of a GPIO



OUTPUT MODE



INPUT MODE

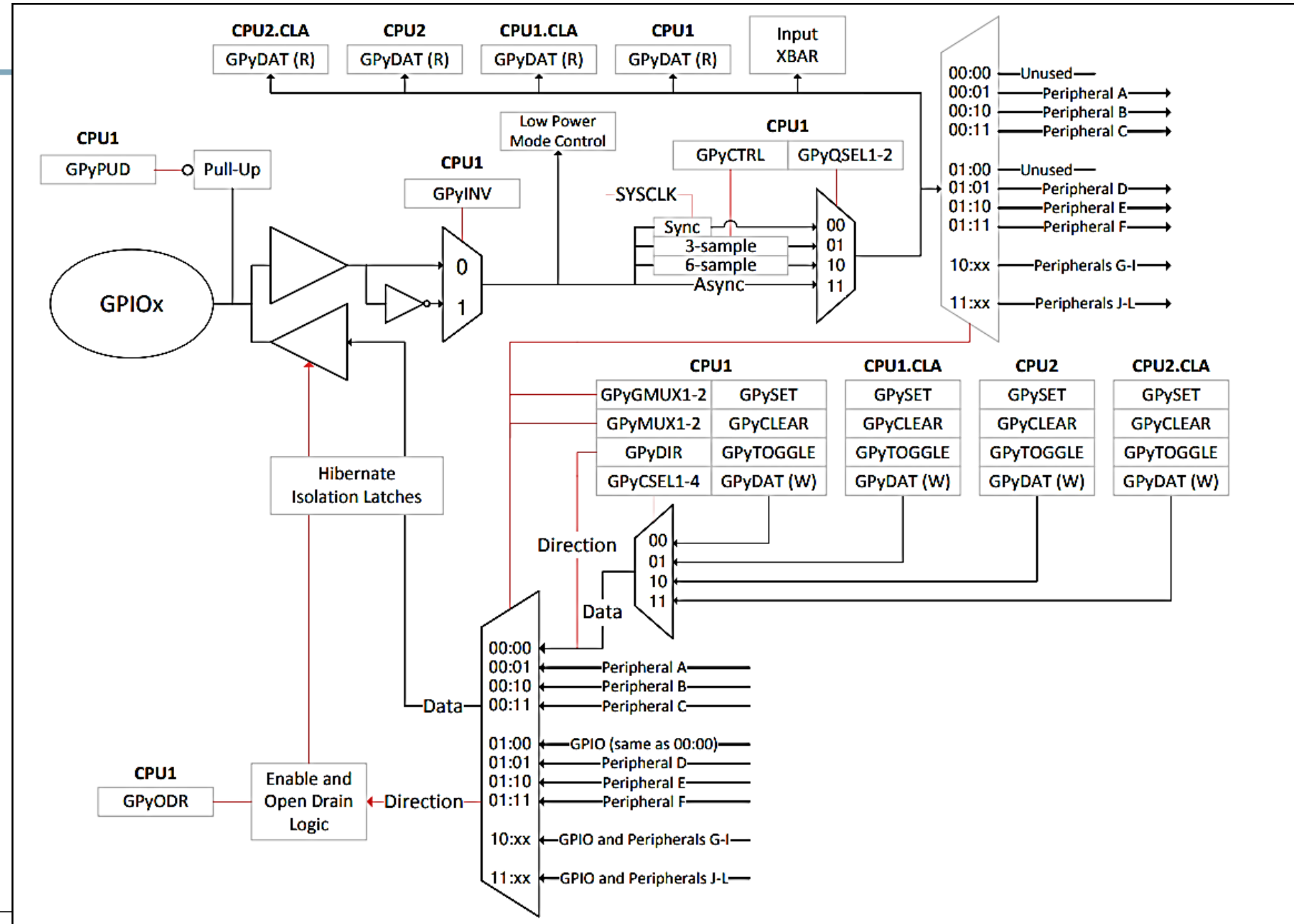
2. GPIO

CPU MASTERS

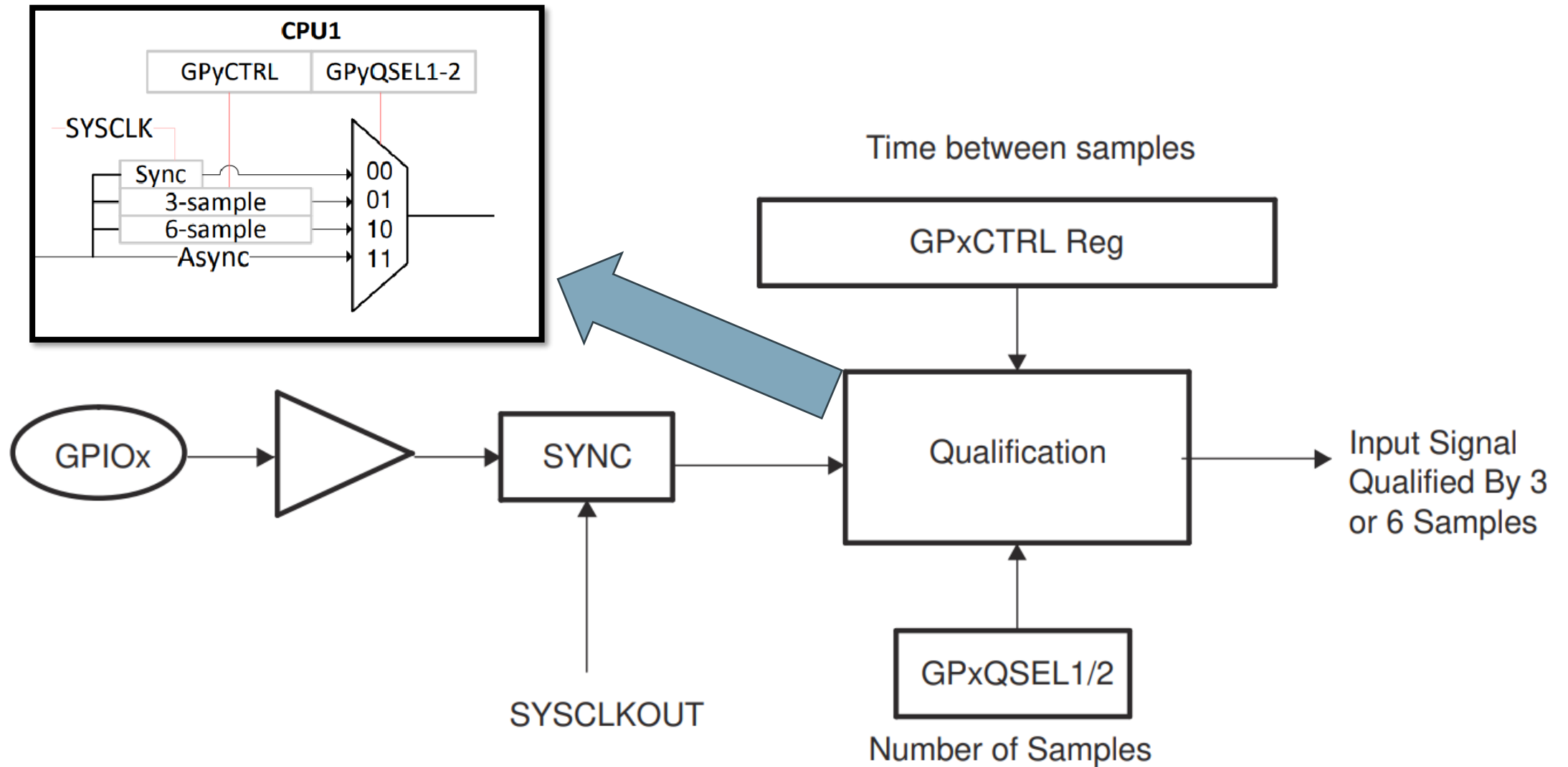
- CPU1
- CPU1.CLA
- CPU2
- CPU2.CLA

There are up to 8 possible I/O ports:

- Port A consists of GPIO0-GPIO31
- Port B consists of GPIO32-GPIO63
- Port C consists of GPIO64-GPIO95
- Port D consists of GPIO96-GPIO127
- Port E consists of GPIO128-GPIO159
- Port F consists of GPIO160-GPIO191
- Port G consists of GPIO192-GPIO223
- Port H consists of GPIO224-GPIO255



3. Input Qualification



3. Input Qualification

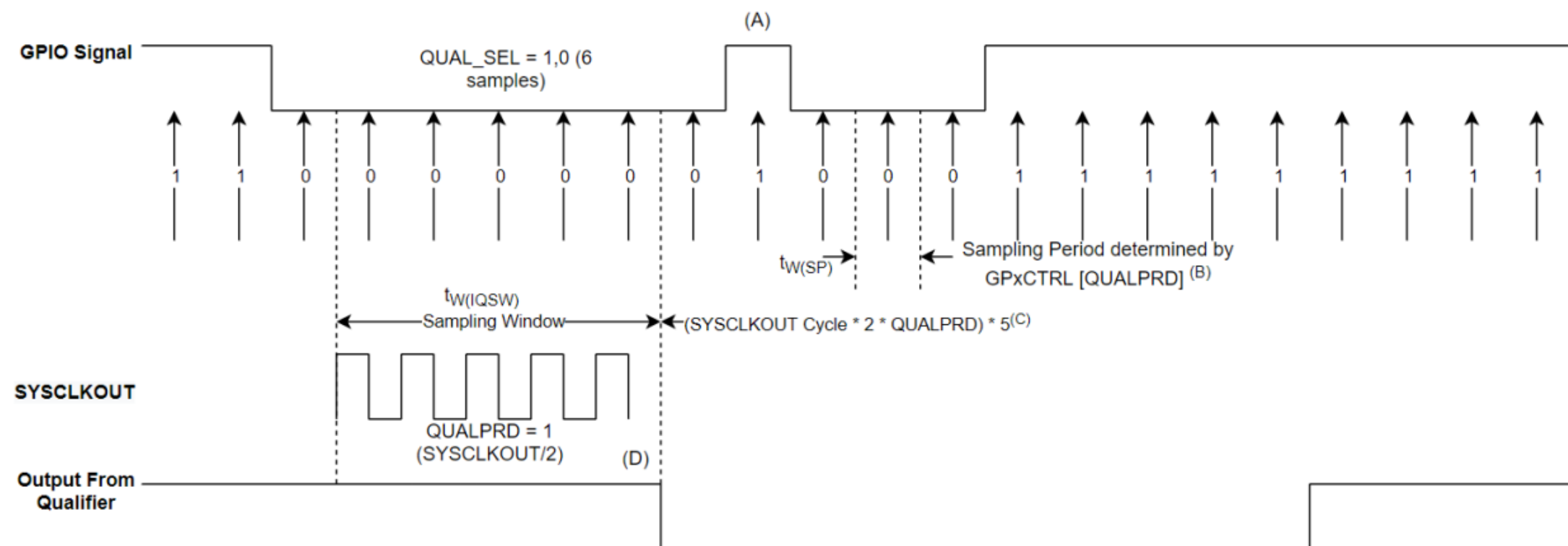


Table 8-4. Case 1: Three-Sample Sampling Window Width

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	$2 \times T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] \neq 0	$2 \times 2 \times GPxCTRL[QUALPRDn] \times T_{SYSCLKOUT}$
Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT	

Table 8-5. Case 2: Six-Sample Sampling Window Width

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	$5 \times T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] \neq 0	$5 \times 2 \times GPxCTRL[QUALPRDn] \times T_{SYSCLKOUT}$
Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT	














4. Init a project

Linked Resources

Path Variables Linked Resources

Path variables specify locations in the file system, including other path variables with the syntax "\${VAR}".
The locations of linked resources may be specified relative to these path variables.



Defined path variables for resource 'TEST':

Name	Value
 C2000WARE_COMMON_INCLUDE	\${COM_TI_C2000WARE_SOFTWARE_PACKAGE_INSTALL_DIR}\device_support\f2837xd\common\include
 C2000WARE_DLIB_ROOT	\${COM_TI_C2000WARE_SOFTWARE_PACKAGE_INSTALL_DIR}\driverlib\f2837xd\driverlib
 C2000WARE_HEADERS_INCLUDE	\${COM_TI_C2000WARE_SOFTWARE_PACKAGE_INSTALL_DIR}\device_support\f2837xd\headers\include
 CCS_BASE_ROOT	C:\ti\ccs1250\ccs\ccs_base
 CCS_INSTALL_ROOT	C:\ti\ccs1250\ccs
 CG_TOOL_ROOT	C:\ti\ccs1250\ccs\tools\compiler\ti-cgt-c2000_22.6.1.LTS
 COM_TI_C2000WARE_SOFTWARE_PACKAGE_INSTALL_DIR	C:\ti\c2000\C2000Ware_5_01_00_00
 ECLIPSE_HOME	C:\ti\ccs1250\ccs\eclipse\
 PARENT_LOC	E:\Drugs\LAUNCHXL_F28379D\F28379D_CCS
 PROJECT_LOC	E:\Drugs\LAUNCHXL_F28379D\F28379D_CCS\TEST
 TI_PRODUCTS_DIR	C:\ti
 TI_PRODUCTS_DIR_TIREX	C:\ti
 WORKSPACE_LOC	E:\Drugs\LAUNCHXL_F28379D\F28379D_CCS

Linked Resources

Path Variables Linked Resources

Linked resources in project 'TEMPLATE_VER_1':

Resource Name	Location
▼  Variable Relative Location	
 driverlib.lib	COM_TI_C2000WARE_SOFTWARE_PACKAGE_INSTALL_DIR\driverlib\f2837xd\driverlib\ccs\Debug\driverlib.lib

4. Init a project

General

Configuration: CPU1_RAM [Active]

Project Products

C2000Ware [5.1.0.00]

Device

Family: C2000

Variant and core: <select or type filter text> TMS320F28379D <any compatible core>

Connection: Texas Instruments XDS100v2 USB Debug Probe

☒ Manage the project's target-configuration automatically

Project type and tool-chain

Compiler version: TI v22.6.1.LTS

Output type: Executable

Output format: legacy COFF

Device endianness: little

Linker command file: 2837xD_RAM_Ink_cpu1.cmd

Runtime support library: <automatic>

4. Init a project

- device
 - device.c
 - device.h
 - driverlib.h
- targetConfigs
 - 2837xD_RAM_Ink_cpu1.cmd
 - F2837xD_CodeStartBranch.asm
 - F2837xD_DefaultISR.c
 - F2837xD_GlobalVariableDefs.c
 - F2837xD_Gpio.c
 - F2837xD-Headers_nonBIOS_cpu1.cmd
 - F2837xD_PieCtrl.c
 - F2837xD_PieVect.c
 - F2837xD_SysCtrl.c
 - F2837xD_usDelay.asm
 - main.c
 - 2837xD_FLASH_Ink_cpu1.cmd
 - driverlib.lib

Include Options

Configuration: CPU1_RAM [Active]

Add dir to #include search path (--include_path, -I)

`$(COM_TI_C2000WARE_SOFTWARE_PACKAGE_INCLUDE_PATH)`

`$(PROJECT_ROOT)`

`$(C2000WARE_COMMON_INCLUDE)`

`$(C2000WARE_HEADERS_INCLUDE)`

`$(PROJECT_ROOT)/device`

`$(C2000WARE_DLIB_ROOT)`

`$(CG_TOOL_ROOT)/include`

File Search Path

Configuration: CPU1_RAM [Active]

Include library file or command file as input (--library, -l)

`$(COM_TI_C2000WARE_SOFTWARE_PACKAGE_LIBRARIES)`

libc.a

Add <dir> to library search path (--search_path, -i)

`$(COM_TI_C2000WARE_SOFTWARE_PACKAGE_LIBRARY_PATH)`

`$(CG_TOOL_ROOT)/lib`

`$(CG_TOOL_ROOT)/include`

Predefined Symbols

Configuration: CPU1_RAM [Active]

Pre-define NAME (--define, -D)

`$(COM_TI_C2000WARE_SOFTWARE_PACKAGE_SYMBOLS)`

_LAUNCHXL_F28379D

DEBUG

_DUAL_HEADERS

CPU1

_LAUNCHXL_F28379D
DEBUG
_DUAL_HEADERS
CPU1

5. Register

GpioCtrlRegs.register / GpioDataRegs.register (lab file: Gpio.c)

Register	Description	
GPxCTRL	GPIO x Control Register	Control
GPxQSEL1	GPIO x Qualifier Select 1 Register	
GPxQSEL2	GPIO x Qualifier Select 2 Register	
GPxMUX1	GPIO x Mux1 Register	
GPxMUX2	GPIO x Mux2 Register	
GPxDIR	GPIO x Direction Register	
GPxPUD	GPIO x Pull-Up Disable Register	
GPxINV	GPIO x Input Polarity Invert Registers	
GPxGSEL1	GPIO x Peripheral Group Mux	
GPxGSEL2	GPIO x Peripheral Group Mux	
GPxCSEL1	GPIO x Core Select Register	
GPxCSEL2	GPIO x Core Select Register	
GPxCSEL3	GPIO x Core Select Register	
GPxCSEL4	GPIO x Core Select Register	
GPxDAT	GPIO x Data Register	Data
GPxSET	GPIO x Data Set Register	
GPxCLEAR	GPIO x Data Clear Register	
GPxTOGGLE	GPIO x Data Toggle Register	

Where x = A, B, C, D, E, or F

Programming Model Comparison

Direct Register Access

- ◆ Register addresses # defined individually
- ◆ User must compute bit-field masks
- ◆ Not easy-to-read

```
*CMPR1 = 0x1234;
```

Bit Field Header Files

- ◆ Header files define all registers as structures
- ◆ Bit-fields directly accessible
- ◆ Easy-to-read

```
EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * duty;
```

DriverLib

- ◆ DriverLib performs low-level register manipulation
- ◆ Easy-to-read
- ◆ Highest abstraction level

```
EPWM_setCounterCompareValue(EPWM2_BASE, EPWM_COUNTER_COMPARE_A, duty);
```

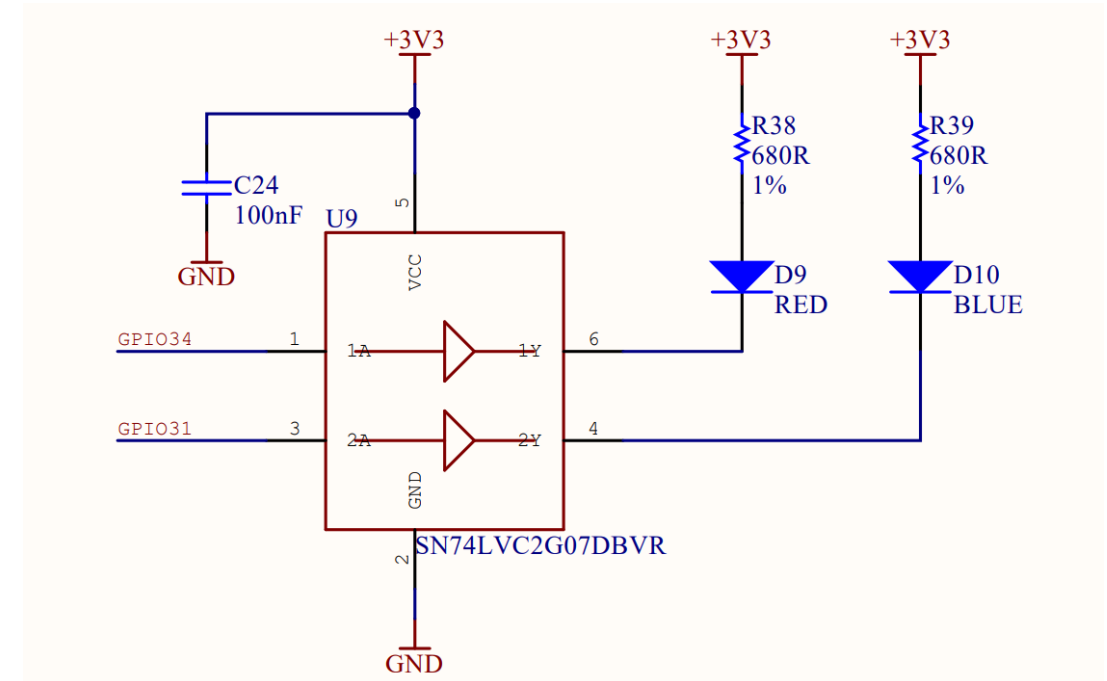
5. Example

Toggle On-board LEDs

```
// BLUE LED
EALLOW;
GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0x0; // GPIO
GpioCtrlRegs.GPAGMUX2.bit.GPIO31 = 0x0; // GPIO
GpioCtrlRegs.GPADIR.bit.GPIO31 = 1; // OUTPUT
GpioCtrlRegs.GPACSEL4.bit.GPIO31 = 0; // CPU1

// RED LED
GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0x0; // GPIO
GpioCtrlRegs.GPBGMUX1.bit.GPIO34 = 0x0; // GPIO
GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; // OUTPUT
GpioCtrlRegs.GPBCSEL1.bit.GPIO34 = 0; // CPU1

GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1; // Toggle BLUE LED
GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Toggle RED LED
DELAY_US(500000);
```



5. Example

Input Qualification * Connect GPIO0 with GPIO1 to simulate the button

```
// GPIO0
GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0x0; // GPIO
GpioCtrlRegs.GPAGMUX1.bit.GPIO0 = 0x0; // GPIO
GpioCtrlRegs.GPADIR.bit.GPIO0 = 1; // OUTPUT
GpioCtrlRegs.GPACSEL1.bit.GPIO0 = 0; // CPU1

// GPIO1
GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0x0; // GPIO
GpioCtrlRegs.GPAGMUX1.bit.GPIO1 = 0x0; // GPIO
GpioCtrlRegs.GPADIR.bit.GPIO1 = 0; // INPUT
GpioCtrlRegs.GPACSEL1.bit.GPIO1 = 0; // CPU1

GpioCtrlRegs.GPACTRL.bit.QUALPRD0 = 0xff; // 255
GpioCtrlRegs.GPACSEL1.bit.GPIO0 = 0x10; // 6 sample

/*
 * Sampling period
 *  $T_{sp} = 2 \cdot \text{QUALPRD} / F_{sys} = 2 \cdot 255 / 200\text{Mhz} = 2.55\mu\text{s}$ 
 * Sampling Window
 *  $T_{sw} = 5 \cdot T_{sp} = 12.75\mu\text{s}$ 
 */

GPIO1_STATE = GpioDataRegs.GPADAT.bit.GPIO1; //
// Press - > 0
if(!GPIO1_STATE){
    GpioDataRegs.GPACLEAR.bit.GPIO31 = 1; // Turn on led
    GpioDataRegs.GPASET.bit.GPIO0 = 1; // Release btn
    DELAY_US(500000);
}else{
    GpioDataRegs.GPASET.bit.GPIO31 = 1; // Turn off led
    GpioDataRegs.GPACLEAR.bit.GPIO0 = 1; // Release btn
    DELAY_US(500000);
}
```

5. Example

Interface with MCP4921 (12-bit DAC) (Using SOFTWARE SPI)

REGISTER 5-1: WRITE COMMAND REGISTER FOR MCP4921 (12-BIT DAC)

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
0	BUF	$\overline{\text{GA}}$	$\overline{\text{SHDN}}$	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
bit 15								bit 0							

bit 15 0 = Write to DAC register
 1 = Ignore this command

bit 14 **BUF:** V_{REF} Input Buffer Control bit
 1 = Buffered
 0 = Unbuffered

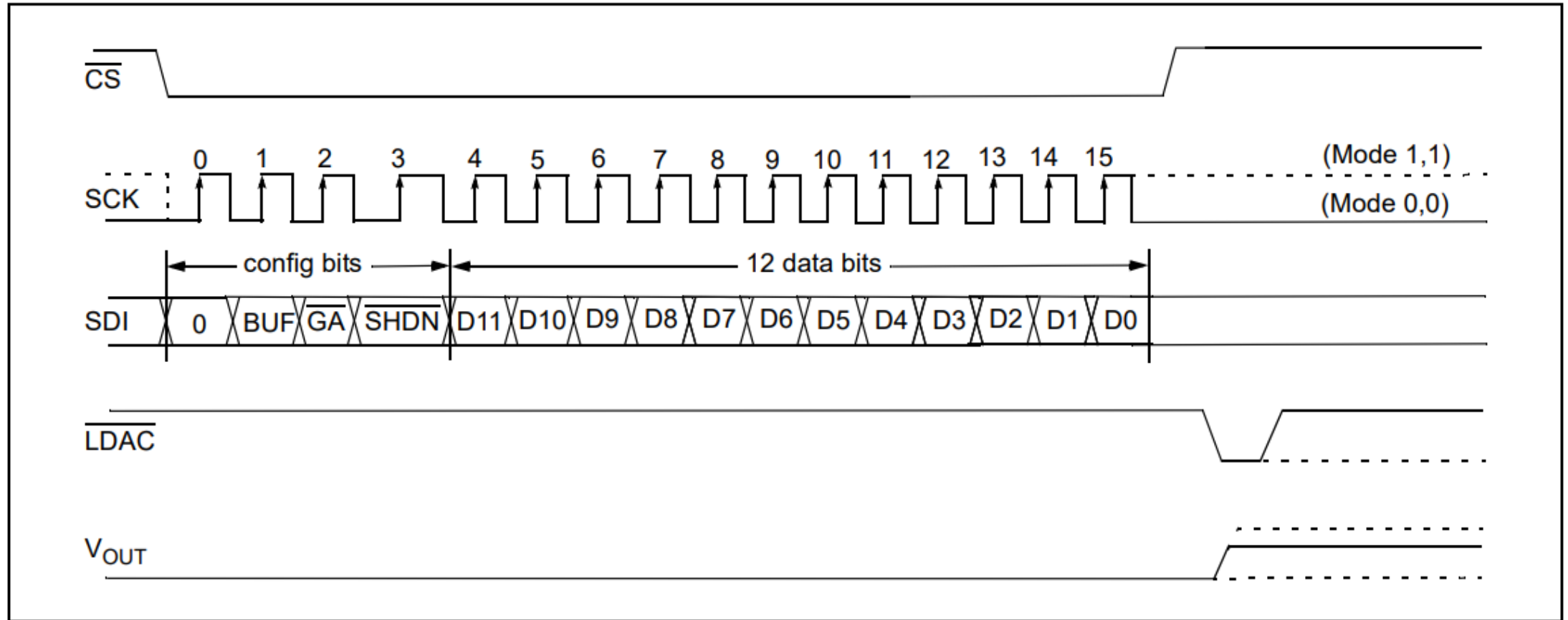
bit 13 **$\overline{\text{GA}}$:** Output Gain Selection bit
 1 = $1\times (V_{\text{OUT}} = V_{\text{REF}} * D/4096)$
 0 = $2\times (V_{\text{OUT}} = 2 * V_{\text{REF}} * D/4096)$

bit 12 **$\overline{\text{SHDN}}$:** Output Shutdown Control bit
 1 = Active mode operation. V_{OUT} is available.
 0 = Shutdown the device. Analog output is not available. V_{OUT} pin is connected to 500 k Ω (typical).

bit 11-0 **D11:D0:** DAC Input Data bits. Bit x is ignored.

5. Example

Interface with MCP4921 (12-bit DAC) (Using SOFTWARE SPI)

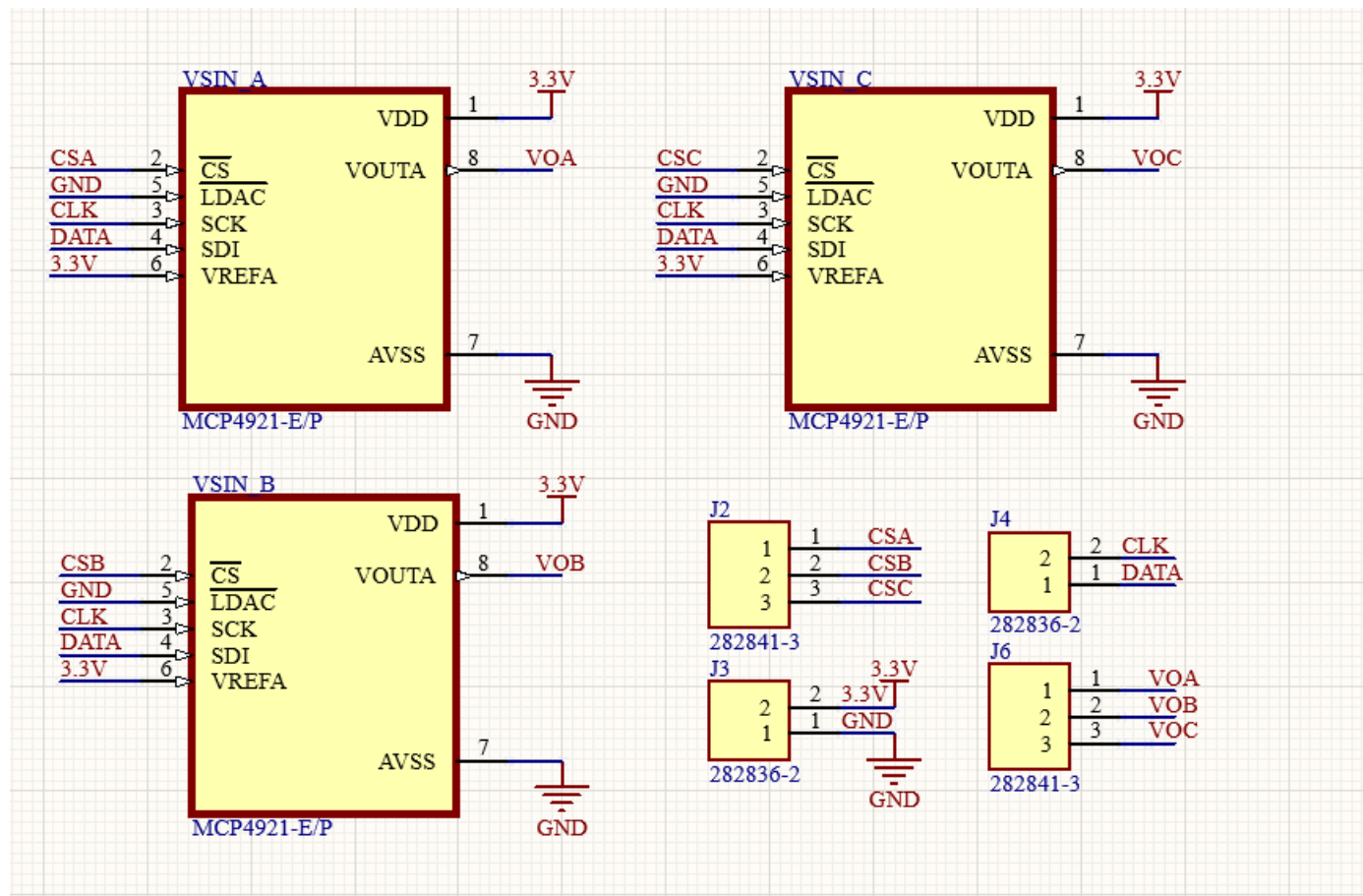


Timing diagram

5. Example

Interface with MCP4921 (12-bit DAC) (Using SOFTWARE SPI)

Main Function (sending 2 bytes using SPI protocol)



Connection schematic

MCP4291	F28379D
CSA	GPIO63
CSB	GPIO64
CSC	GPIO26
CLK	GPIO27
DATA	GPIO25

5. Example

Interface with MCP4921 (12-bit DAC) (Using SOFTWARE SPI)

Main Function (sending 2 bytes using SPI protocol)

```
void SEND_DAC(uint16_t value){
    value = (value&0xffff)|0x7000;
    uint16_t __ibit;
    for(__ibit = 0x8000 ; __ibit > 0; __ibit>>=1){
        DATA(__ibit&value);
        CLK(1);CLK(0);
    }
}
```

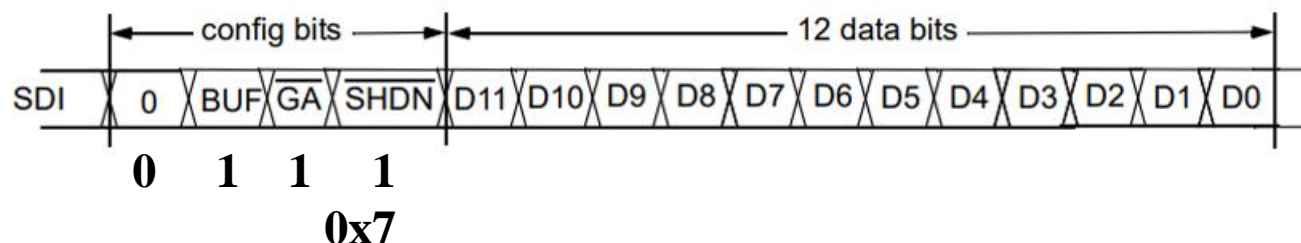
bit 15 0 = Write to DAC register
 1 = Ignore this command

bit 14 **BUF**: V_{REF} Input Buffer Control bit
 1 = Buffered
 0 = Unbuffered

bit 13 **GA**: Output Gain Selection bit
 1 = $1\times (V_{OUT} = V_{REF} * D/4096)$
 0 = $2\times (V_{OUT} = 2 * V_{REF} * D/4096)$

bit 12 **SHDN**: Output Shutdown Control bit
 1 = Active mode operation. V_{OUT} is available.
 0 = Shutdown the device. Analog output is not available. V_{OUT} pin is connected to 500 k Ω (typical).

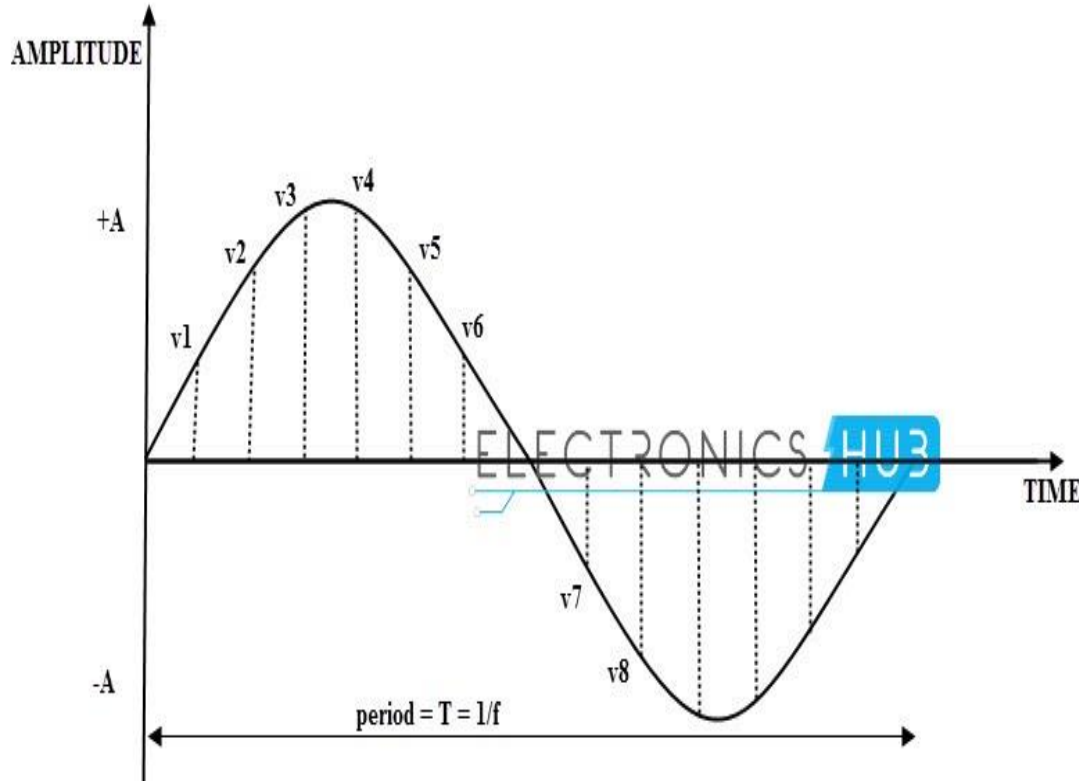
bit 11-0 **D11:D0**: DAC Input Data bits. Bit x is ignored.



5. Example

Interface with MCP4921 (12-bit DAC) (Using SOFTWARE SPI)

Generate Sine waveform using MCP4921



Now if u want to create a sine waveform with the fundamental frequency is 50Hz

Suppose the waveform is created with **200 frames per period**.
Meaning in 20ms u have to update the waveform 200 times.

Let do a simple math here:

20ms -> 200 frames

1 frame -> 0.1ms = 100us

USE DELAY_US(100);

Should Use Interrupt

5. Example

Interface with MCP4921 (12-bit DAC) (Using SOFTWARE SPI)

Generate Sine waveform using MCP4921



Delay by
delay
function

Delay by
sending
process

T_delay

Sendac

100us

xxus

> 100us

Fundamental Frequency is not 50 Hz

THANKS FOR WATCHING



HCMUTE

Monna Dang (Dang Hoang Anh Chuong)

02/2024

Some Useful Links

Resource Explorer (Online course)

https://dev.ti.com/tirex/explore/node?node=A__AEd34C6EIRh7UOzlmPh7Fg__c2000ware_software_package__gYkahfz__LATEST

https://software-dl.ti.com/C2000/docs/optimization_guide/intro.html

CCS Guide

https://software-dl.ti.com/ccs/esd/documents/users_guide/index.html

DESIGNDRIVE

<https://www.ti.com/tool/DESIGNDRIVE>

FAQ

<https://e2e.ti.com/support/microcontrollers/c2000-microcontrollers-group/c2000/f/c2000-microcontrollers-forum>

SinTable GEN

<https://ppelikan.github.io/drlut/>