OpAmp Programming

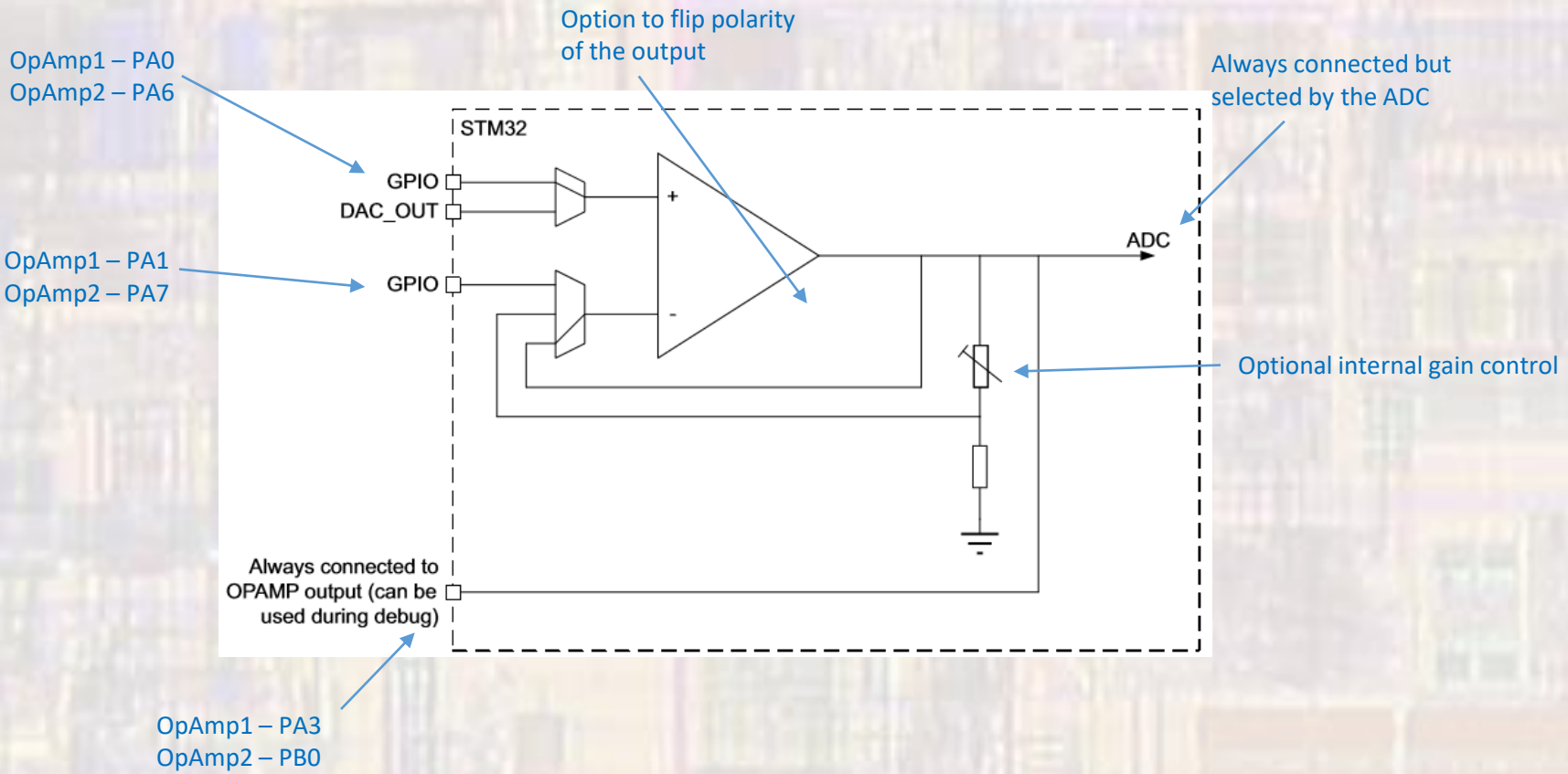
Last updated 6/23/21

OpAmp Programming

- ADC Resolution
  - Nucleo-L476RG has two OpAmps
    - Mbed does not support the OpAmps
    - We will need to write our own low-level code
  - Our board does not allow us to use D0/D1
    - We only have access to OpAmp2

OpAmp Programming

- OpAmp Configuration



OpAmp Programming

- OpAmp Connections
  - Nucleo-L476RG VINM, VINP, and VOUT connections are fixed
  - Multiple steps are required to access the connections
    - Port(s) must be enabled (clocks enabled) – PortA is enabled by default
    - Pins must be selected as Analog Inputs (using the Pin I/O configuration registers)

Table 150. Operational amplifier possible connections

Signal	Pin	Internal	comment
OPAMP1_VINM	PA1 or dedicated pin(1)	OPAMP1_OUT or PGA	controlled by bits OPAMODE and VM_SEL.
OPAMP1_VINP	PA0	DAC1_OUT1	controlled by bit VP_SEL.
OPAMP1_VOUT	PA3	ADC1_IN8 ADC2_IN8	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.
OPAMP2_VINM	PA7 or dedicated pin(1)	OPAMP2_OUT or PGA	controlled by bits OPAMODE and VM_SEL.
OPAMP2_VINP	PA6	DAC1_OUT2	controlled by bit VP_SEL
OPAMP2_VOUT	PB0	ADC1_IN15 ADC2_IN15	The pin is connected when the OPAMP is enabled. The ADC input is controlled by ADC.

1. The dedicated pin is only available on BGA132 and BGA169 (for STM32L49x/L4Ax devices) package. This configuration provides the lowest input bias current (see datasheet).

OpAmp Programming

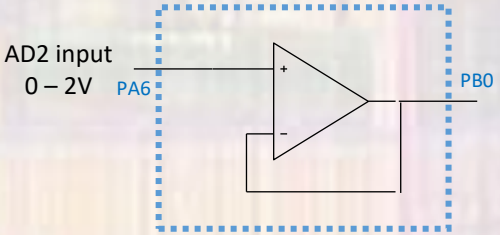
- Simple example 1
  - OpAmp setup - follower

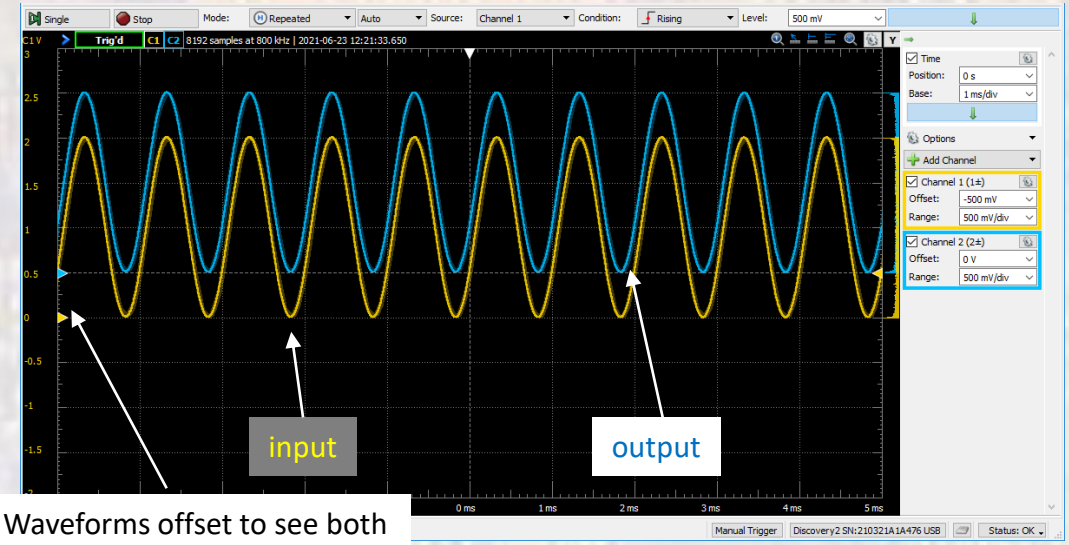
```
////////////////////////////////////
//
// opamp_class_ex_1 project
//
// created 6/4/21 by tj
// rev 0
//
////////////////////////////////////
//
// OpAmp example file for class
//
// shows basic opamp operation
// also shows direct register access
//
////////////////////////////////////

#include "mbed.h"
//#include <stdio.h>

int main(void){
    printf("RCC: %x\n", RCC->AHB2ENR);
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("opamp_class_ex_1 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
           MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // Using OPAMP2: PA6 as VINP, PA7 as VINM, PB0 as the output

    // Must enable PortA to access the opamp inputs
    // RCC_AHB2ENR bit 0 for port A
    RCC->AHB2ENR |= 0x01;
    // Set VINP(PA6) and VINM(PA7) to analog in mode - 11 for bits 15-14 and 13-1
    GPIOA->MODER |= 0x00000F000;

    // Must enable PortB to access the opamp output (??)
    // RCC_AHB2ENR bit 1 for port B
    RCC->AHB2ENR |= 0x02;
```

```
    // Enable opamp peripheral clock (common for both)
    // APB1 (high speed APB)
    // RCC_APB1ENR1, 1 to bit 30 to enable
    RCC->APB1ENR1 |= 0x40000000;

    // Setup opamp values - follower mode
    // VINP to GPIO, 0 to bit 10
    // VINM - not used in this mode
    // Follower mode - 11 to bits 3-2
    // all others 0
    OPAMP2->CSR = 0x0000000C;

    // Enable opamp
    // enable - 1 to bit 0
    OPAMP2->CSR |= 0x00000001;

    while(1){
        // Nothing to do here
        // Printing to make sure we are operating
        printf("still running\n");

        wait_us(1000000);
    }

    return 0;
}// end main
```

OpAmp Programming

- Simple example 1 - results
  - OpAmp setup - follower





OpAmp Programming

- Simple example 2
  - OpAmp setup – programmable gain

```
////////////////////////////////////
//
// opamp_class_ex_2 project
//
// created 6/4/21 by tj
// rev 0
//
////////////////////////////////////
//
// OpAmp example file for class
//
// shows basic opamp operation - gain mode
// also shows direct register access
//
////////////////////////////////////

#include "mbed.h"
//#include <stdio.h>

int main(void){
    printf("RCC: %x\n", RCC->AHB2ENR);
    setbuf(stdout, NULL);   // disable buffering

    // splash
    printf("opamp_class_ex_2 - example for EE2905\n");
    printf("Using Mbed OS version %d.%d.%d\n\n",
            MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);

    // Using OPAMP2: PA6 as VINP, PA7 as VINM, PB0 as the output

    // Must enable PortA to access the opamp inputs
    // RCC_AHB2ENR bit 0 for port A
    RCC->AHB2ENR |= 0x01;
    // Set VINP(PA6) and VINM(PA7) to analog in mode - 11 for bits 15-14 and 13-12
    GPIOA->MODER |= 0x00000F000;

    // Must enable PortB to access the opamp output (??)
    // RCC_AHB2ENR bit 1 for port B
    RCC->AHB2ENR |= 0x02;
```

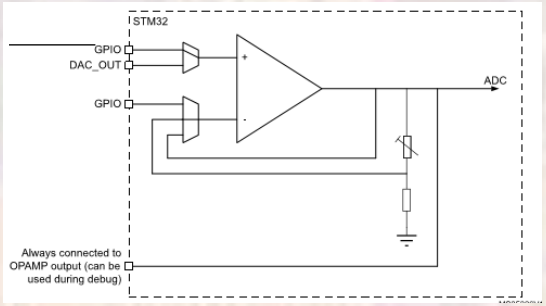
```
    // Enable opamp peripheral clock (common for both)
    // APB1 (high speed APB)
    // RCC_APB1ENR1, 1 to bit 30 to enable
    RCC->APB1ENR1 |= 0x40000000;

    // Setup opamp values - internal gain mode
    // VINP to GPIO, 0 to bit 10
    // VINM - not used in this mode
    // Internal gain 2-00, 4-01, 8-10, 16-11 bits 5-4
    // internal gain mode - 10 to bits 3-2
    // all others 0
    // start with gain=2
    OPAMP2->CSR = 0x00000008;

    // Enable opamp
    // enable - 1 to bit 0
    OPAMP2->CSR |= 0x00000001;

    while(1){
        // Cycle throug varius gain values
        // Assume a 10KHz input (100us period)
        // Try to get 5 cycles at each gain
        // Printing to make sure we are operating
        OPAMP2->CSR = 0x00000009;
        wait_us(500);
        OPAMP2->CSR = 0x00000019;
        wait_us(500);
        OPAMP2->CSR = 0x00000029;
        wait_us(500);
        OPAMP2->CSR = 0x00000039;
        wait_us(500);
    }
    return 0;
}// end main
```

OpAmp Programming

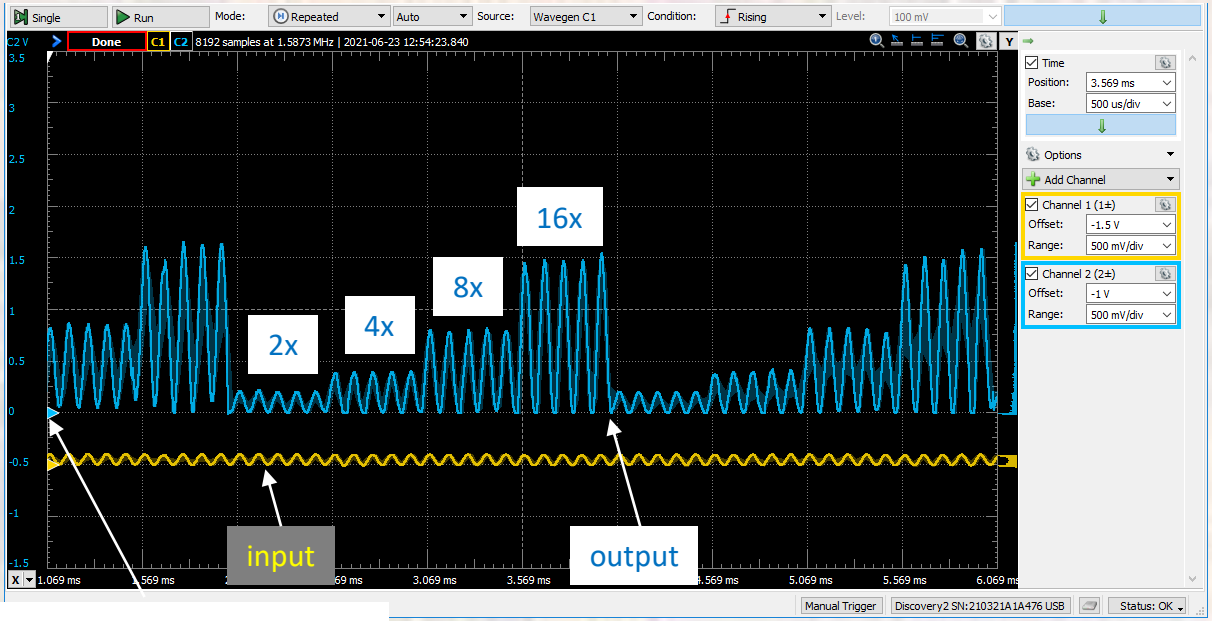
- Simple example 2 - results
  - OpAmp setup – programmable gain

AD2 input
0 – 100mV

PA6



PB0



Waveforms offset to see both

OpAmp Programming

- Limitations
  - Only OpAmp 2 available to us
  - 3mV input offset spec
  - 1.6MHz Gain-Bandwidth