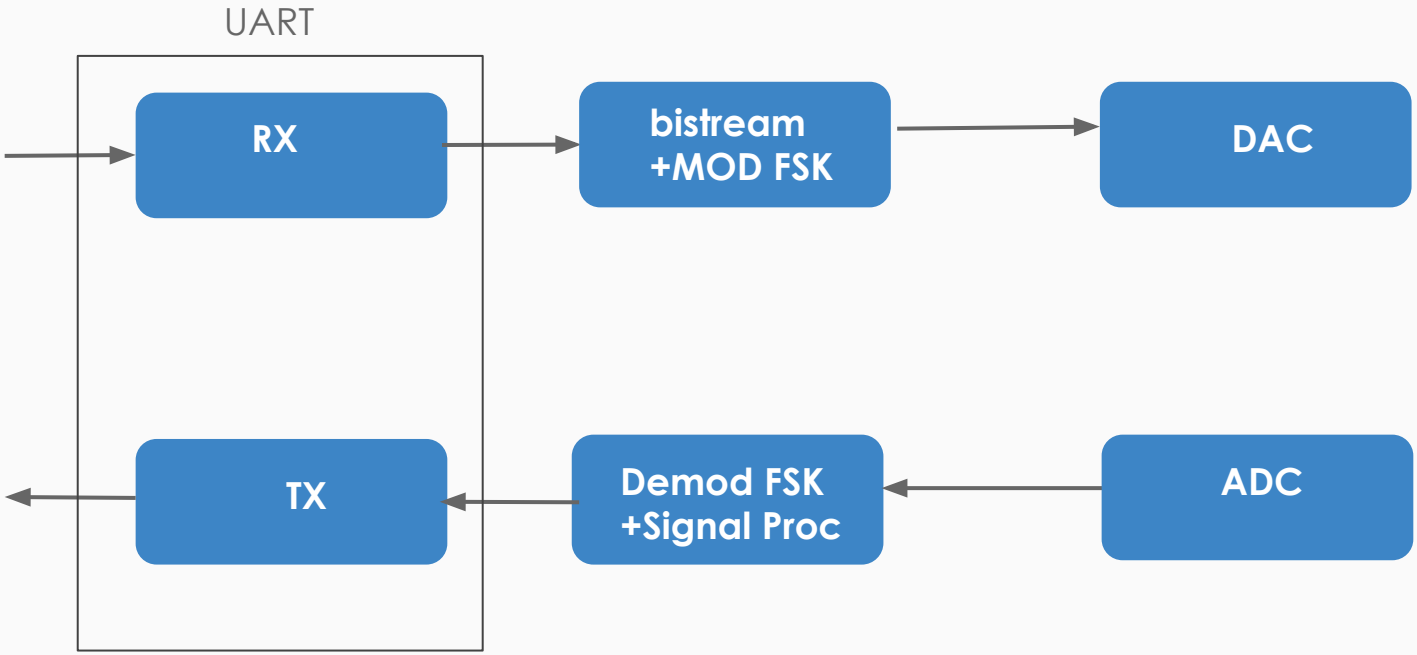
Laboratorio de Microprocesadores

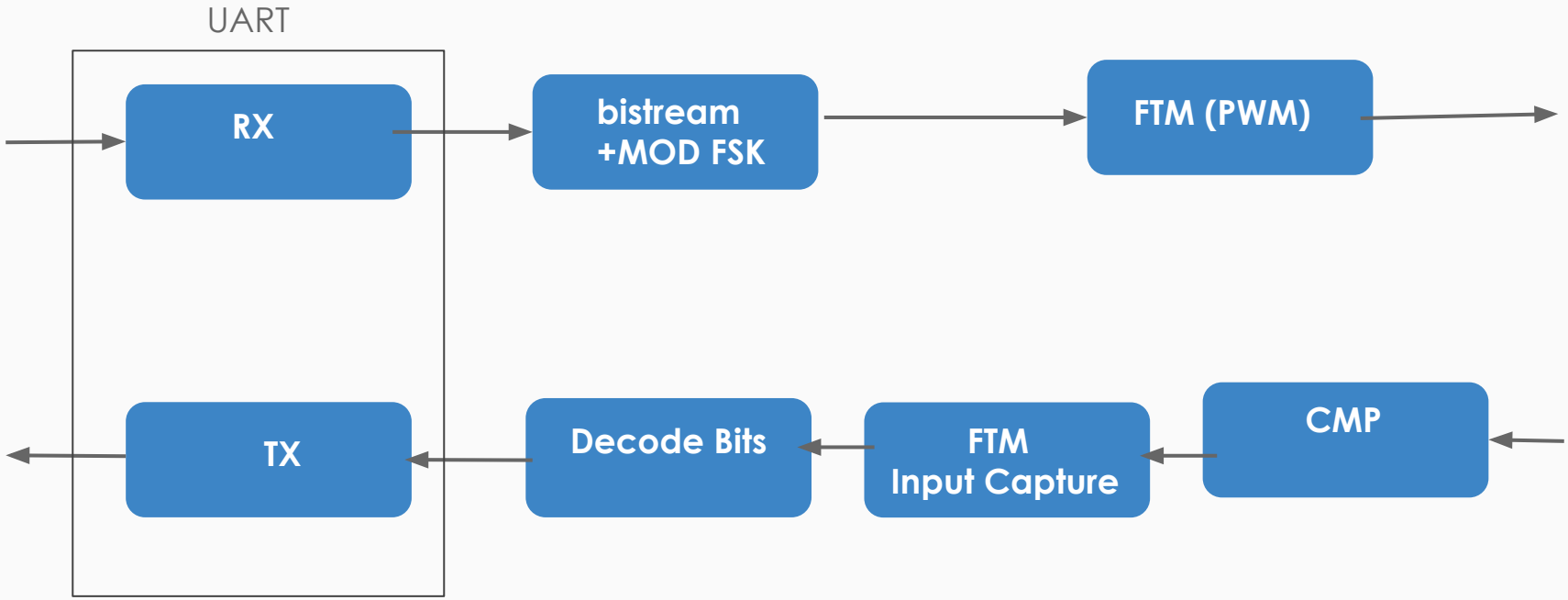
Trabajo Práctico №3

ITBA

Máspero Martina
Mestanza Joaquín
Nowik Ariel
Regueira Marcelo



Versión 1 del Trabajo Práctico



Versión 2 del Trabajo Práctico

Capas e interfaces

Application

APP

Hardware Abstraction Layer

DEMOD (V1/V2) MOD(V1/V2)

MicroController Abstraction Layer

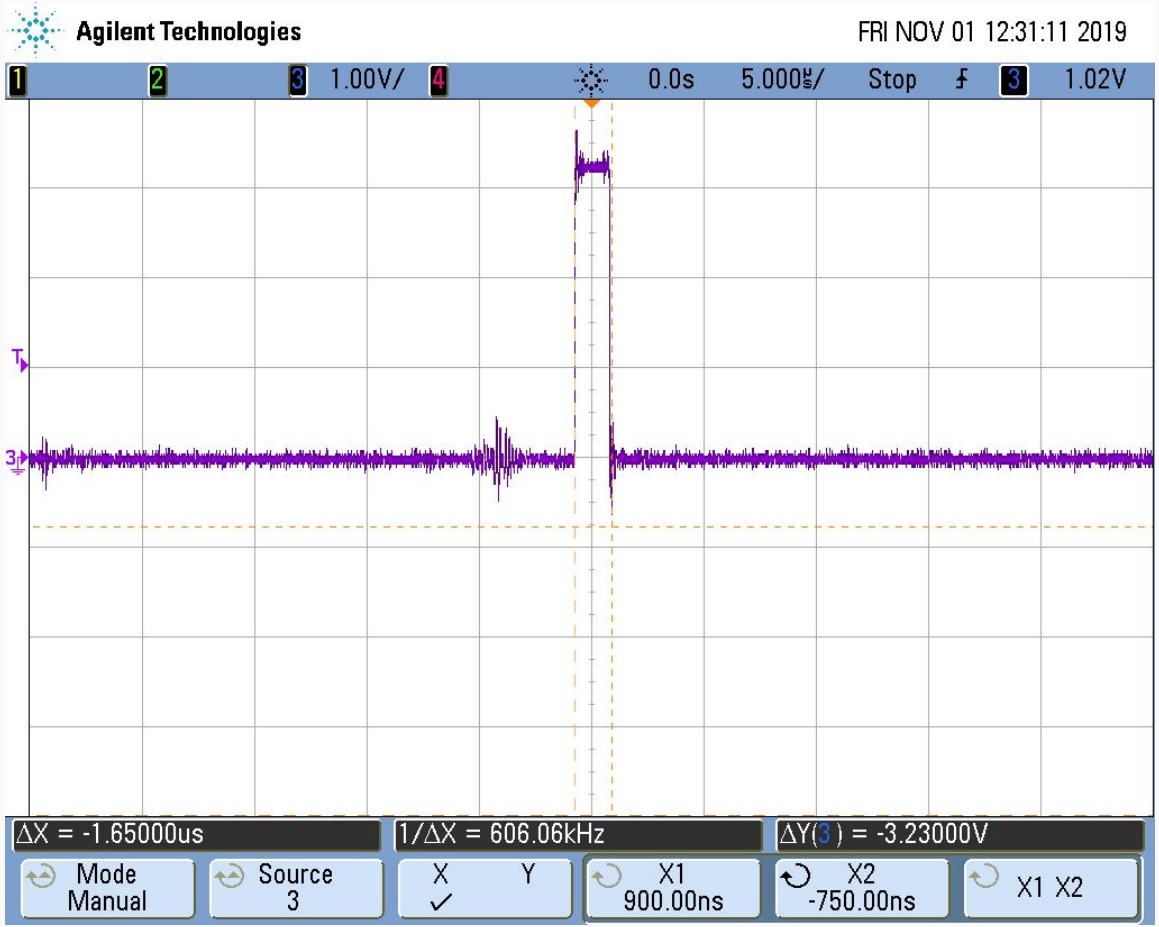
GPIO ADC DAC CMP PIT DMA FTM UART

UART

Se agregó la recepción. Se utiliza mediante dos funciones, updateWord() que procesa los nuevos caracteres recibidos y setOnNewCharListener para establecer qué debe suceder cuando llega un nuevo caracter.

UART.h

```
void uartInit(uart_cfg_t config);

void sendWord(const char *word);

void updateWord();
void setOnNewCharListener(void *recv(char));
```



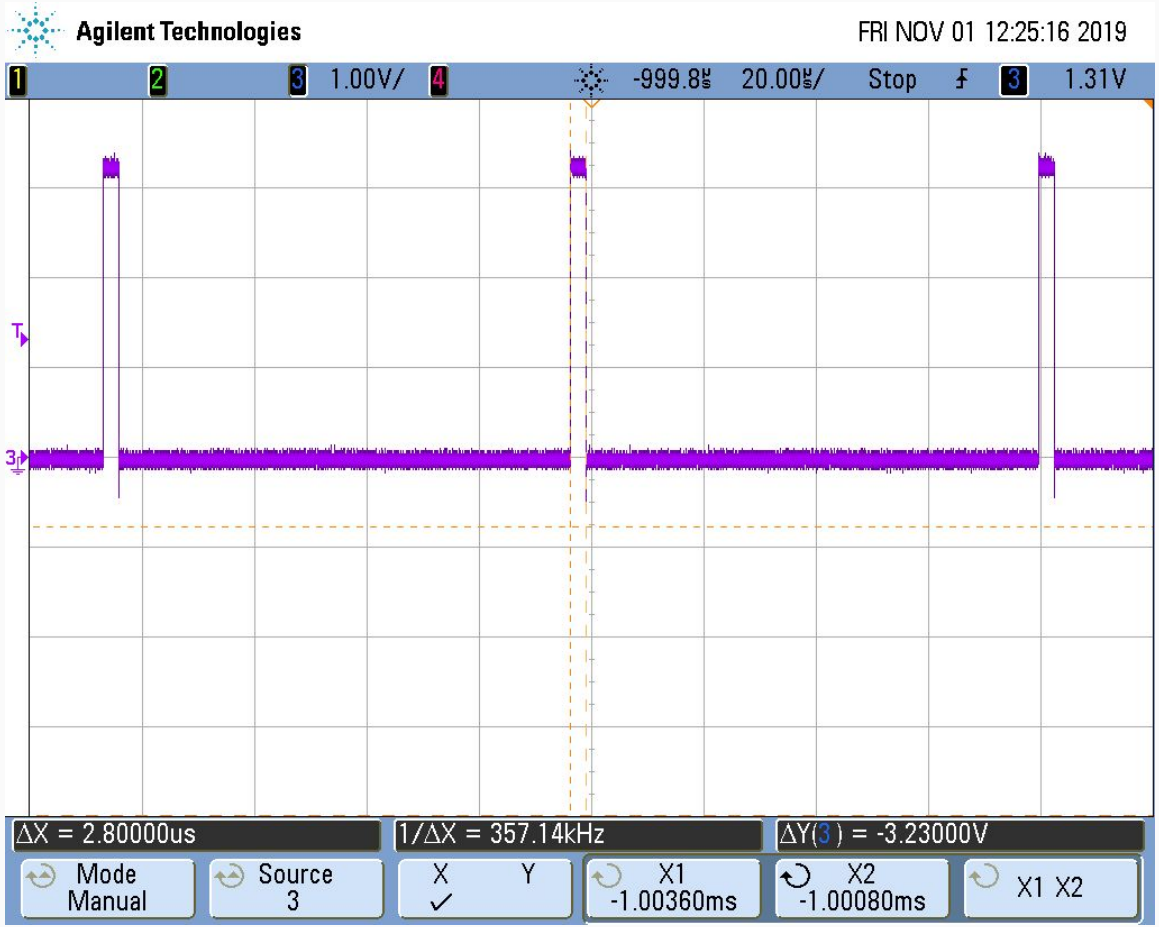
Tiempo de ISR para UART

ADC

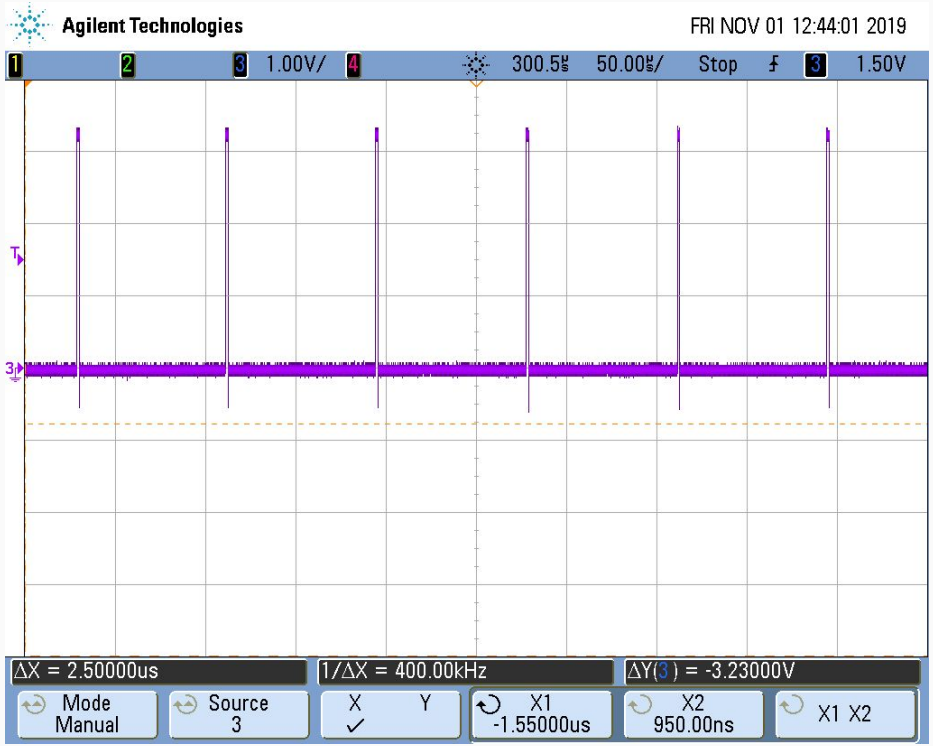
El ADC se encuentra conectado al DAC. Está configurado para triggerearse por software llamando a startConvertion.

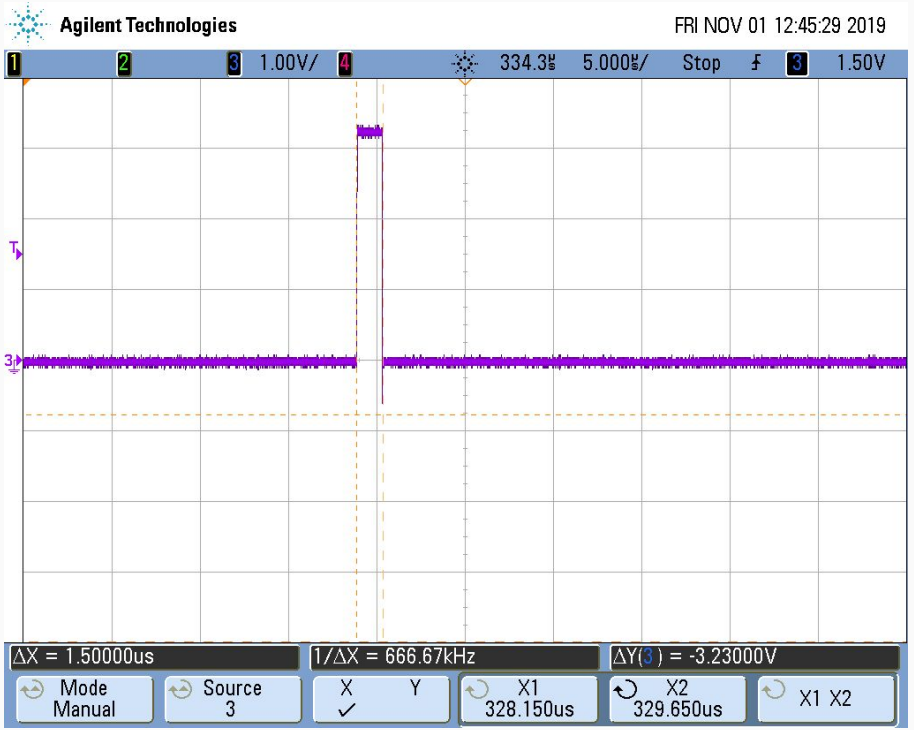
ADC.h

```
void ADC_init(uint32_t adc_ch, void (*funcallback)(void));

void ADC_enableModule(uint32_t adc_ch);

void ADC_disableModule(void);

ADCResultData_t ADC_getDataResult(void);

void ADC_startConvertion (void);
```



Tiempo de ISR del ADC (llamada a callback)





Tiempo de ISR de llamada de PIT al ADC (tiempo de conversión)

DAC

Se dispara con la función de DAC_setData, donde se pasa un valor de 12 bits que representa un valor analógico entre 0V y 3.3V.

DAC.h

```
void DAC_init(void);

void DAC_setData(DACDATA_t data);
```

Dem. FSK

En el Init se configura al PIT que inicia las conversiones del ADC. Cada vez que hay un nuevo valor se guarda y se hace el cálculo para demodular. Una vez que hay un mensaje listo se llama al callback que se pasa como parámetro en el Init.

demoduladorFSK.h

```
void FSKdem_init( myCallback
funcallback);
char get_Msg (void);
void FSKdemodulate(void);
bool isDataReady(void);
```

Mod. FSK

Se llama a sendChar con el caracter que se desea modular, donde se arma el paquete agregando START, PARIDAD y STOP, y se pasa al DAC. Al terminar se llama al callback.
Mientras no hay nada para enviar se mantiene en estado IDLE mandando '1'.
En el INIT se configura el PIT que define cada cuanto tiempo se llama al DAC.

Modulador.h

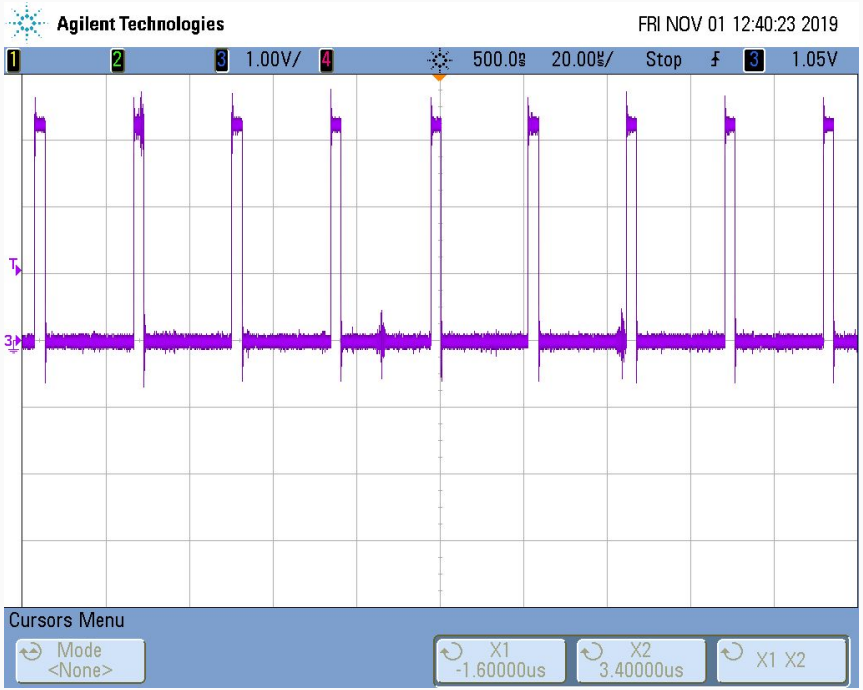
```
void Modulador_init(void(*funcallback)(void));

void Modulador_sendChar(char my_char);
```

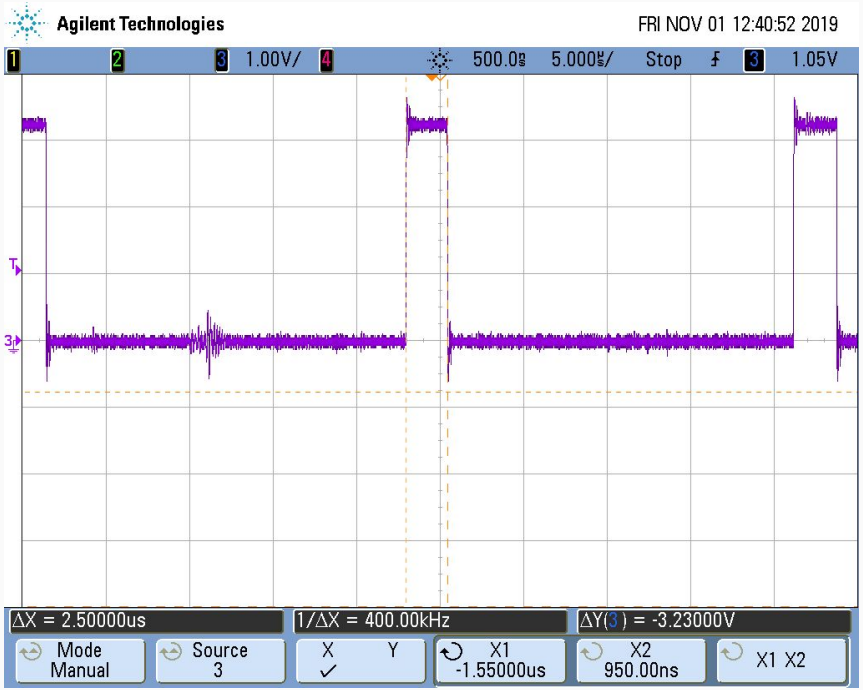
PIT

Timer que se utiliza para regular los tiempos del modulador para el envío de las señales en la versión 1 (los valores del arreglo que forman la senoidal). Permite configurar más de uno y se puede iniciar o detener con las funciones correspondientes.

PIT.h

```
void PIT_init(void);

void PIT_configTimer(uint8_t id, uint16_t value,
void(*funcallback)(void));

void PIT_startTime(uint8_t timer_id);

void PIT_stopTime(uint8_t timer_id);
```





Tiempo de ISR de PIT para actualizar DAC

FTM

En este driver se implementaron las siguientes funcionalidades:

- Input Capture
- Output Compare
- PWM
- Edge Aligned
- Center Aligned

Las funcionalidades para las segunda versión utilizadas fueron:

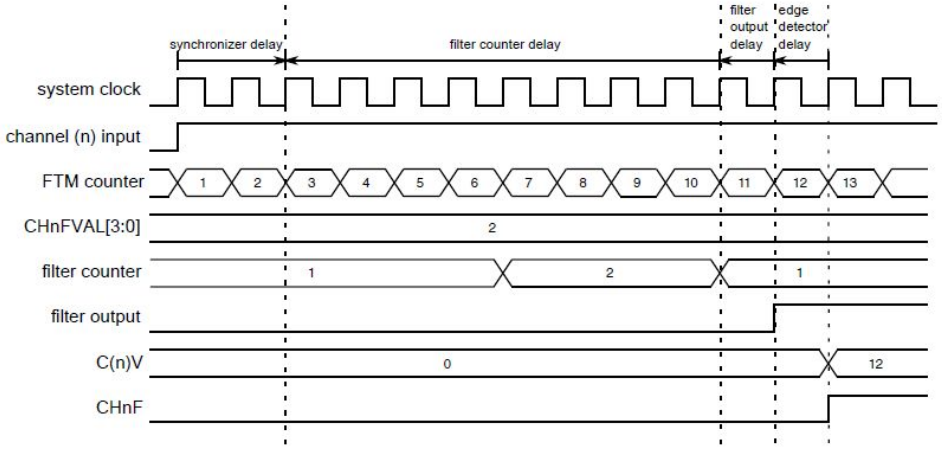


Figure 40-14. Input capture example

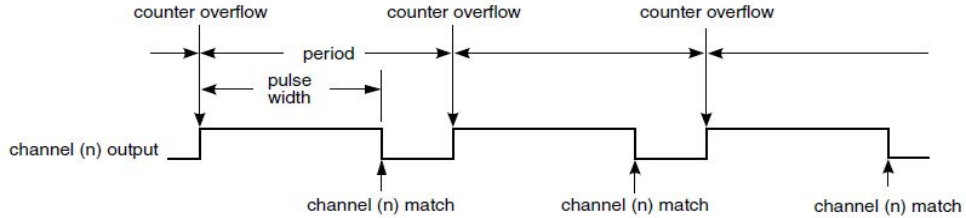
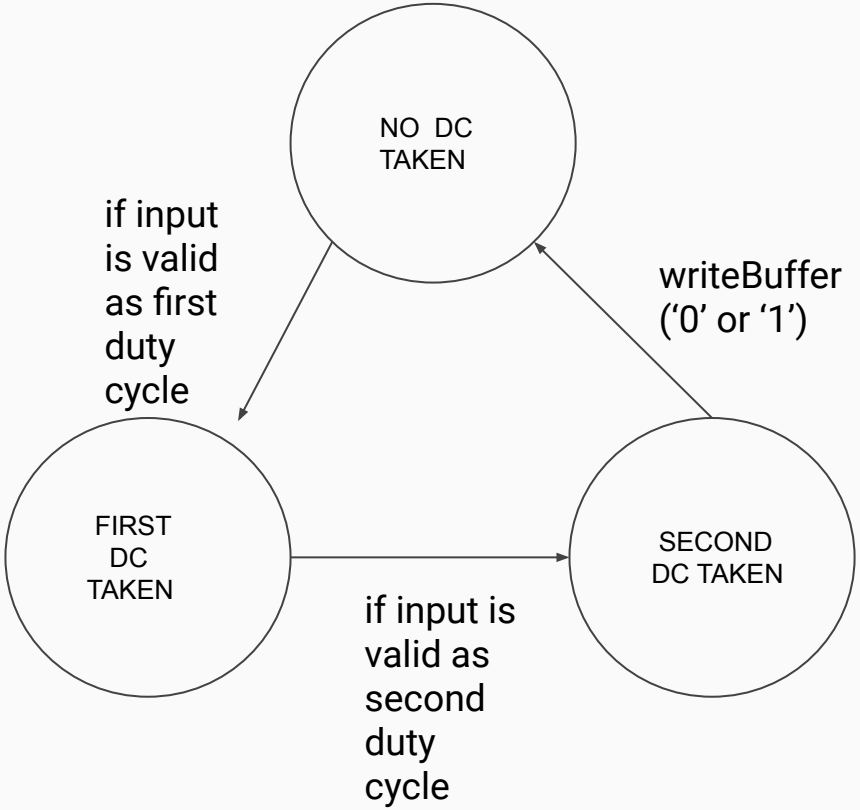


Figure 40-18. EPWM period and pulse width with ELSnB:ELSnA = 1:0

Decode Bits

Obtenidas las mediciones de input capture, esta función lo que hace es según el dato que llega, se lo guarda en un buffer circular.
Los datos que llegan pueden ser 0%, 50% o 100% Duty Cycle.



CMP

El comparador se utiliza en la versión 2 para pasar la senoidal a una cuadrada, que luego se utilizará como entrada en el input capture.
Solo requiere inicializarse indicando si se utiliza el 0 o el 1 (en nuestro caso usamos el 0)

CMP.h

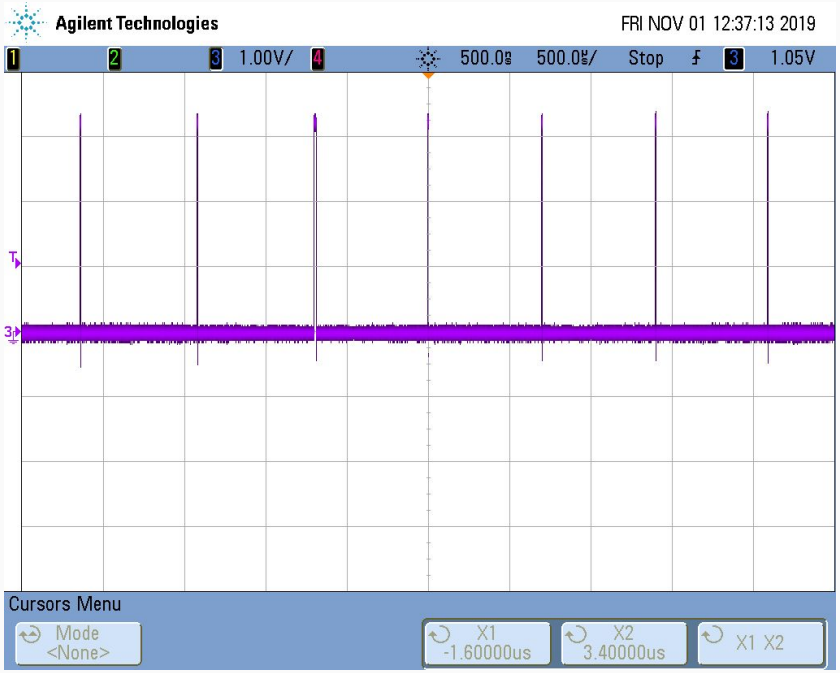
void CMP_init(cmps_ids id);

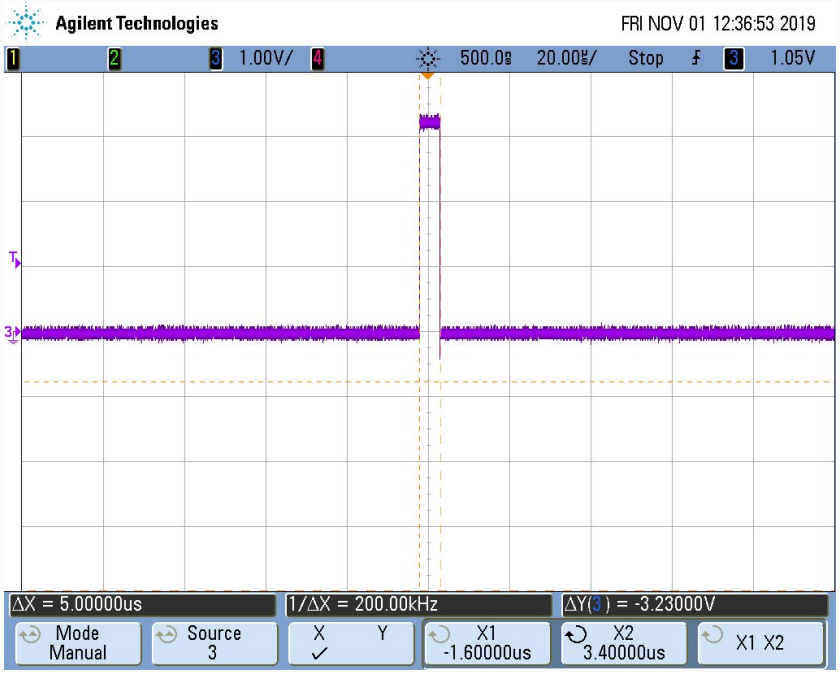
DMA

Se utilizó en la versión 2. Se hacen las configuraciones básicas con Config.
El channel 0 se utilizó para mover datos hacia el PWM, y el channel 1 para mover datos desde el Input Capture.

DMA.h

```
void DMA0_Config(uint16_t *source_add, uint16_t* dest_add,
void(*funcallback)(void));
void DMA1_Config(uint32_t *source_add, uint32_t* dest_add,
void(*funcallback)(void));
void DMA0_ConfigCounters(uint8_t channel, uint32_t
source_full_size, uint32_t source_unit_size);
void DMA0_ConfigSourceAddress(uint8_t channel, uint32_t
*source_add);
void DMA0_ConfigDestAddress(uint8_t channel, uint32_t
*dest_add);
void DMA0_EnableRequest(uint8_t channel);
void DMA0_DisableRequest(uint8_t channel);
```





Tiempo de ISR de DMA Handler para el Modulador Versión 2

¿ Preguntas ?