

# **Trabajo Integrador Final**

Sánchez Fernando y Boattini Nicolás.  
Noviembre 2019.

Universidad de la Cuenca del Plata.  
Facultad de Ingeniería-Ingeniería en Sistemas.  
Arquitectura de Computadores

En las siguientes páginas de este documento, se dará introducción y detalles sobre el trabajo integrador final realizado para la cátedra de arquitectura de computadores del segundo cuatrimestre del primer año de Ingeniería en Sistemas, en la Universidad de la Cuenca del Plata. En dicho trabajo se realiza una implementación de varios conceptos aprendidos a lo largo del cursado, en un solo sistema. Se dispondrá de todo el material utilizado, bibliografías consultadas, fichas técnicas y sitios web revisados, respectivamente, permitiendo al lector la máxima comprensión e inmersión dentro del proyecto llevado a cabo.

Capítulo 1 Introducción .....	1
Introducción .....	1
La respuesta .....	1
Capítulo 2 El Proyecto .....	2
Metodología .....	2
La sintaxis .....	4
Montaje y Programación .....	5
Lista de Materiales .....	6
Capítulo 3 Marco Teórico .....	7
Compuertas lógicas .....	7
Pasaje Binario a Decimal .....	8
Capítulo 4 Análisis y Conclusión. ....	9
Análisis .....	9
Conclusión .....	9
Bibliografía y Webgrafía .....	10
ANEXOS .....	11
Imágenes .....	11

## **Capítulo 1**

### **Introducción**

#### **Introducción**

El Objetivo del proyecto realizado, es implementar conceptos, técnicas, métodos y hardware aprendidos, en un solo sistema. Tales como, el uso y funcionamiento de un display de 7 segmentos, conversión de sistema decimal a binario, funcionamiento de determinados componentes electrónicos, programación en Arduino con C, implementación del lenguaje Assembler (Assembly/ Lenguaje de Ensamblador) y compuertas lógicas (simuladas por código).

#### **La respuesta**

Una vez propuesto lo anterior, es el momento de aplicarlo a un sistema. Nuestra respuesta es un prototipo electrónico, que consta de una placa ArduinoUNO, programada en C, con partes embebidas en Assembler, un display 7 segmentos, 1 sensor infrarrojo, 5 leds, 5 resistencias de 330 Ohms y 30 cables.

Este prototipo, muestra a través del display 7 segmentos una secuencia repetida de números decimales del 0 al 9 y a la par va mostrando en un grupo de 4 leds que representan 4 bits, la correspondencia del número decimal mostrado en el 7 segmento, en binario.

Luego dispone de un cambio de función de función por código, se constan de 4 funciones en total. La primera función es la principal, que son las acciones anteriormente

mencionadas, las otras 3 corresponden a la simulación de compuertas lógicas mediante código, estas compuertas simuladas son la OR, AND y XOR, y la salida es otro led.

## **Capítulo 2**

### **El Proyecto**

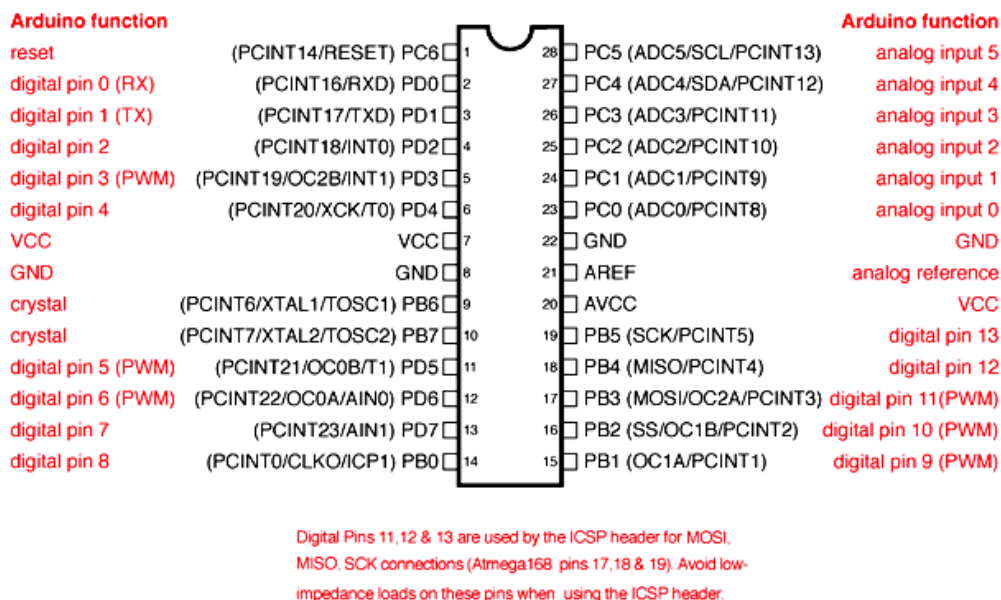
#### **Metodología**

La realización del proyecto, en principio fue una tarea muy desafiante, debido a la propuesta de implementar líneas de código en Lenguaje de Ensamblador. Analizando todo lo propiamente dicho a realizar, se decidió comenzar por la búsqueda del Lenguaje de Ensamblador, o Assembler/Assembly, ya que era la rama desconocida de todo el proyecto.

Se dedicaron horas y horas a la búsqueda de ejemplos de código en Assembler, prototipos con Arduino en Assembler, manuales, funciones, como opera, qué tipos de dato utiliza, que datos recibe, cuales devuelve, cuáles son las sentencias, como se finaliza una línea, etc. También comprendimos que, para utilizar los pines del Arduino, debíamos hacer referencia al pin del microcontrolador ATMEGA328p que se corresponde con la placa de Arduino. Para nuestra sorpresa, cada grupo de pines están en un puerto, y para acceder a cada pin, debíamos acceder primero a su registro, y luego al pin, por lo cual debíamos conocer el mapeado de pines. *Imagen 2-1*

Imagen 2-1

## ATMega328P and Arduino Uno Pin Mapping



(Mapeado de Pines ATMega328p)

Podemos entender entonces que los pines 0 a 7 están en el Puerto D, los pines 8 a 13 en el Puerto B, y los pines analógicos en el Puerto C.

Una vez conocidos los pines debíamos saber cómo establecerlos como salida y cambiarles su estado. Para esto existen dos registros, el **DDRX** y el **PORTX**, donde X hace referencia a la letra que corresponde a cada puerto. El registro DDR, se puede leer/escribir para establecer/leer si el pin es una entrada o una salida (INPUT/OUTPUT). El registro PORT, se puede leer/escribir para establecer/leer el voltaje de un pin (HIGH/LOW). También existe un tercer registro **PINX**, que solamente lee el estado de un pin declarado como entrada.

Luego, era necesario saber qué sentencia utilizar. Es dónde aparecen *SBI* (Set Bit in I/O Register), y *CBI* (Clear Bit in I/O register). *SBI*, pone a set (1), el estado de un bit de un registro. Por ejemplo, para poner establecer como *OUTPUT*, el pin 13 (*PB5*), deberíamos utilizar *SBI DDRB5*. *CBI*, hace un clear, borra un bit de un registro. Si quisiéramos poner en *LOW* el pin 12, usaríamos *CBI PORTB4*.

Pero no es tan sencillo, existen distintos tipos de funciones en Assembler, la que requerimos son las “*volatile*”, que manipulan datos de entrada, para producir salidas.

### La sintaxis

Para las funciones de Assembler son de la siguiente manera:

```
asm asm-qualifiers ( AssemblerTemplate
                    : OutputOperands
                    [ : InputOperands
                    [ : Clobbers ] ])
```

AssemblerTemplate: va el string, el texto literal con el código ensamblador.

OutputOperands: variables con datos de salida obtenidos en la ejecución del código.

InputOperands: variables con datos de entrada a utilizarse en la ejecución del código.

Clobbers: listas/registros que se pueden utilizar.

En el `AssemblerTemplate`, iría la función *SBI* o *CBI*, la que requiramos utilizar, y las variables, que se enumeran, con %0, %1..., dependiendo de cuantas variables se utilicen. El fin de línea viene dado por “\n\t”, luego le siguen “:”, para dar enumerar variables de salida, pero en nuestro caso no se requiere. Nuevamente “:” y se procede a brindar las entradas, a las que hacen referencia %0, %1..., debemos que hacer referencia al puerto, y al pin que queremos afectar de dicho puerto, entonces en primera instancia tendríamos “T” (`_SFR_IO_ADDR(PORTB)`), haciendo referencia al puerto, “T” notifica al

compilador que devuelva un tipo de dato “puerto/pin”, y luego le seguiría *"I"* (*DDB5*), haciendo referencia al pin 5, del puerto B. Entonces, nuevamente, si quisiéramos poner en HIGH el pin 13 de Arduino con Assembler deberíamos escribir:

```
asm volatile (
    "SBI %0, %1 \n\t"
    :: "I" (_SFR_IO_ADDR(PORTB)), "I" (PORTB5)
    );
```

## Montaje y Programación

Habiendo aprendido las funciones que íbamos a utilizar, decidimos comenzar a montar nuestro prototipo, lo demás ya resultaría más fácil. En primer lugar, se procedió a conectar el display 7 segmentos. Poseemos un FYS-5612A/BX-XX, que es de cátodo común, la ficha técnica de dicho display estará disponible en los anexos.

Una vez conectado el display, realizamos las correspondientes funciones para encenderlo de acorde los números 0 al 9 en secuencia. Para ello, creamos dos funciones en Assembler para que apaguen y enciendan cada uno de los segmentos. Luego creamos otra que recibe el pin y el estado que queremos ponerle. Y todo esto lo controlamos en un switch, que va avanzando de caso en caso mediante una variable que se incrementa al inicio de la ejecución del programa. Cuando esa variable llega a 9, los leds parpadean, y se inicia nuevamente el conteo. Este switch, para cada número, prende o apaga todos los leds, y apaga o enciende los que necesite para mostrar el número deseado, según sea más conveniente.



Siguiente a esto, tenemos el cambio de funciones el cual lo realizamos mediante un sensor. Al detectar un cambio el sensor, el 7 segmentos se pone en “1”, y los leds se apagan. Por pantalla se activa el modo de compuertas. Se solicita que ingrese el valor de A, luego el valor de B, y se selecciona la operación (compuerta) a ejecutar. Dependiendo de cada valor, y de la compuerta seleccionada, se encenderá o no, un led a modo de “salida”. Después de mostrar el resultado, la ejecución se retoma al conteo. Mientras que, por otro lado, la conversión a binario de decimal, se realiza a la par del conteo principal, de una manera sencilla, obteniendo el binario de cada número decimal en 4 bits, y encendiendo el led correspondiente a cada valor de cada bit.

### **Lista de Materiales**

Materiales	Cantidades
ArduinoUNO	1
Protoboard	1
Cables	Aproximadamente 30
Resistencias	5
Leds	5
Sensor Infrarrojo	1
Display 7 Segmentos	1

## Capítulo 3

### Marco Teórico

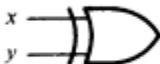
#### Compuertas lógicas

“Las puertas lógicas son circuitos electrónicos que operan con una o más señales de entrada para producir una señal de salida. Las señales eléctricas como voltaje o corriente existen en todas las partes de un sistema digital en cada uno de los dos valores definidos. Los circuitos que operan con voltajes responden a dos rangos separados de voltajes que representan una variable binaria igual a un 1 lógico o a un 0 lógico. Los terminales de entrada de las puertas lógicas aceptan señales binarias dentro del rango permitido y responden a los terminales de salida con señales binarias que caen dentro de un rango específico. Las regiones intermedias entre los rangos permitidos de la figura se cruzan solamente durante los cambios de 1 a 0 o de 0 a 1. A estos cambios se le llaman transiciones, y las regiones intermediarias se llaman regiones de tránsito.”

(Fundamentos de diseño Lógico- M. Mano).

**Tablas de verdad para las tres operaciones lógicas básicas**

AND			OR			NOT	
$X$	$Y$	$Z = X \cdot Y$	$X$	$Y$	$Z = X + Y$	$X$	$Z = \bar{X}$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Excluyente-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x$	$y$	$F$
			0	0	0
			0	1	1
			1	0	1
			1	1	0

## Pasaje Binario a Decimal

*“En el sistema decimal se emplean diez dígitos diferentes para representar números en base diez. En el sistema binario se tienen solo los dígitos 1 y 0. Por tanto, los números en el sistema binario se representan en base 2.*

*Para evitar confusiones pondremos a veces un subíndice en los números que indica su base. Por ejemplo,  $83_{10}$  y  $4728_{10}$  son números representados en notación decimal, o simplemente números decimales. El 1 y el 0 en notación binaria tienen el mismo significado que en notación decimal.*

*Para representar números mayores, como ocurre en la notación decimal, cada dígito de un número binario tiene un valor que depende de su posición:*

$$10_2 = (1 \times 2^1) + (0 \times 2^0) = 2_{10}$$

$$11_2 = (1 \times 2^1) + (1 \times 2^0) = 3_{10}$$

$$100_2 = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$$

“

*(“Organización y arquitectura de computadores”. William Stallings)*

## **Capítulo 4**

### **Análisis y Conclusión.**

#### **Análisis**

A pesar de la dificultad propuesta en utilizar Assembler, lo demás ya fue un ámbito conocido para nosotros. Y tratando de implementar varios temas de la cátedra, logramos obtener un buen resultado. Podemos destacar que el hecho de utilizar Assembler, hace que nuestro programa, se ejecute de manera más rápida, y sea más liviano, ya que controlamos una determinada acción, de manera directa, o casi directa.

#### **Conclusión**

Al final de todo, nos llevamos un nuevo conocimiento, el lenguaje Assembler, gracias a esta experiencia integradora mediante la cual pudimos también incorporar nuevos conocimientos, como también volver a ver conceptos ya aprendidos. Si bien este prototipo, de momento, no tiene un fin práctico en la vida cotidiana, varias de sus partes podemos verlas implementadas por todos lados, todos los días, las compuertas, los menús, también la representación binaria y decimal. Y gracias a la materia, y todos los conceptos aprendidos durante el cursado, es que llegamos a plantearnos realizar y logramos concluir este proyecto

### **Bibliografía y Webgrafía**

PUERTOS Y REGISTROS (INGLÉS) -

<https://www.arduino.cc/en/Reference/PortManipulation>

MAPEADO DE PUERTOS - <https://www.incb.com.mx/index.php/articulos/78-microcontroladores-y-dsps/2546-conociendo-el-microcontrolador-nucleo-core-atmega328p-de-arduino-uno-mic019s>

DELAY Y SETEO DE PINES Y ESTADOS (INGLÉS)-

[https://retrobench.fandom.com/wiki/Programming\\_Arduino\\_in\\_Assembly#cite\\_note-0](https://retrobench.fandom.com/wiki/Programming_Arduino_in_Assembly#cite_note-0)

ASSEMBLER EN C (INGLÉS) - <https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html>

EL REGISTRO PORT (ESPAÑOL) - <http://panamahitek.com/registro-port-puerto/>

SENTENCIA CBI (ESPAÑOL)

[http://www.sc.ehu.es/sbweb/webcentro/automatica/web\\_avr/archivos/SET%20AT90S8515/Bit&Test\\_Bit/CBI.htm](http://www.sc.ehu.es/sbweb/webcentro/automatica/web_avr/archivos/SET%20AT90S8515/Bit&Test_Bit/CBI.htm)

SENTENCIA SBI (ESPAÑOL)

[http://www.sc.ehu.es/sbweb/webcentro/automatica/web\\_avr/archivos/SET%20AT90S8515/Bit&Test\\_Bit/SBI.htm](http://www.sc.ehu.es/sbweb/webcentro/automatica/web_avr/archivos/SET%20AT90S8515/Bit&Test_Bit/SBI.htm)

“Organización y arquitectura de computadores”. WILLIAM STALLINGS. Pearson Educación. – Apéndice A, Sistema Binario y Conversión de Binario a Decimal. Páginas 725 a 729.

“Fundamentos De Diseño Logico Y Computadoras” - M. MORRIS MANO & CHARLES R. KIME - PRENTICE HALL.- Capítulo 2, Páginas 29 a 39- Compuertas Lógicas

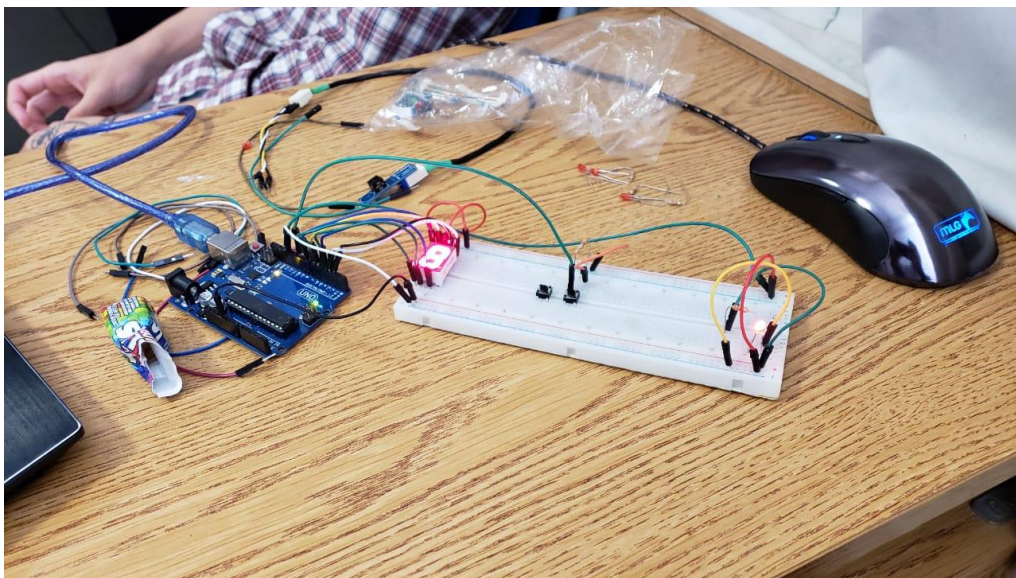
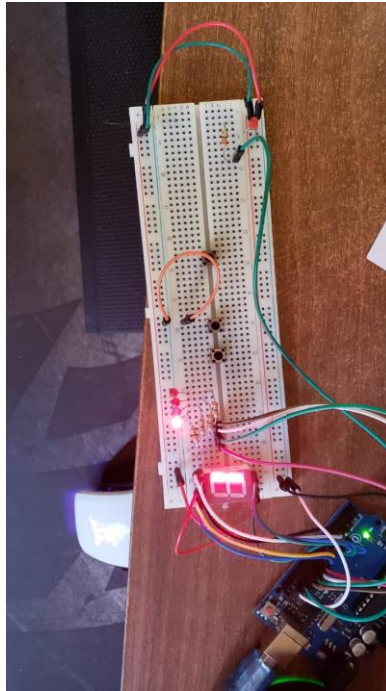
“Arquitectura de Computadoras/Diseño Digital”- M. MORRIS MANO. Capítulo 2, Página 57- Compuertas Lógicas.

MENU Y COMPUERTAS LÓGICAS -

<https://www.youtube.com/watch?v=tK9whAqr9Tk>

## ANEXOS

### Imágenes



# PRODUCT SPECIFICATION

**Model No.: FYS-5612A/BX-XX**

## Descriptions & Features:

- 0.56 Inch (14.20mm) digit height Single Digit Display
- Case mold type.
- RoHS compliant.
- Low current operation
- Low power consumption.
- Easy mounting on P.C. board or socket.



CUSTOMER APPROVED SIGNATURES	APPROVED BY	CHECKED BY	PREPARED BY

**NINGBO FORYARD OPTOELECTRONICS CO.,LTD**

**Add:** NO.115 Qixin Road Ningbo Zhejiang China

**Zip:** 315051

**Tel:** 0086-574-87933652 87927870 87922206

**Fax:** 0086-574-87927917

**E-mail:** Sales@foryard.com (General)

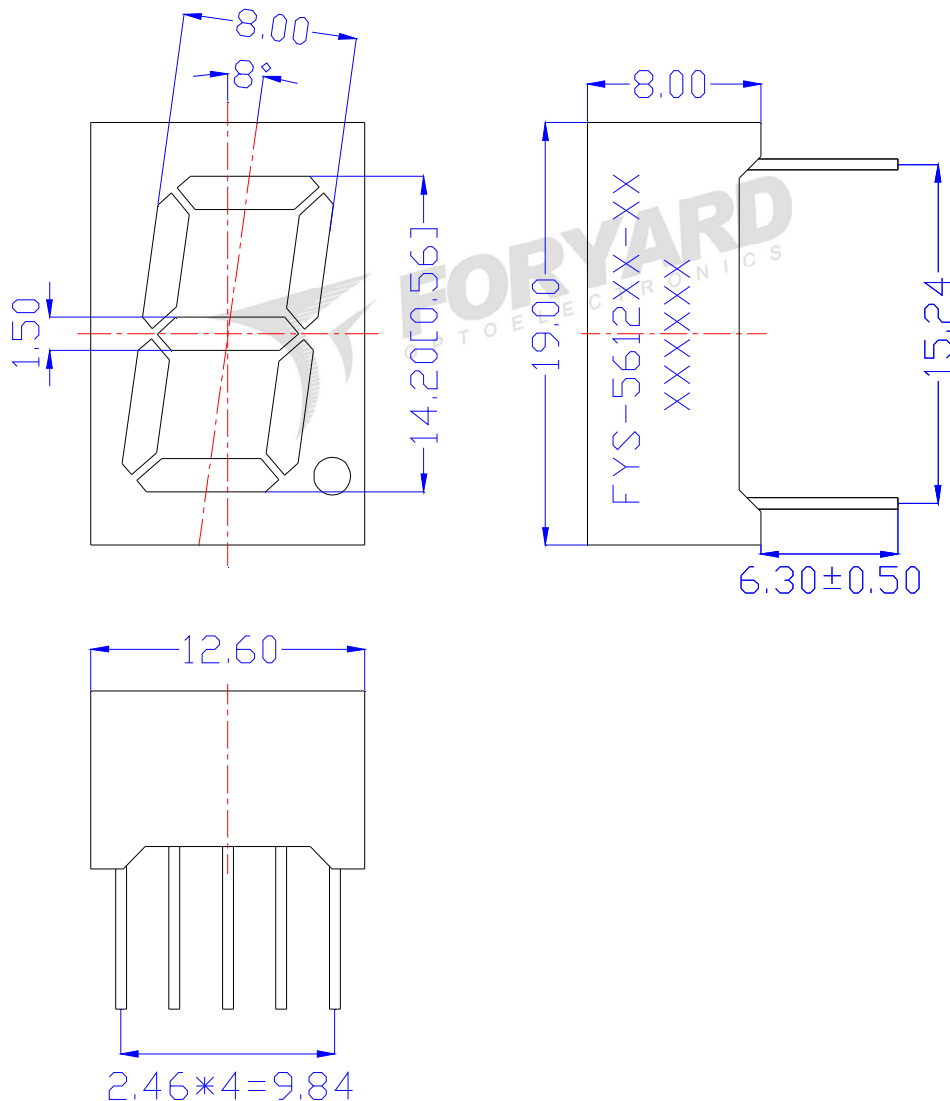
**Http://**[www.foryard.com](http://www.foryard.com)

Model No.: FYS-5612A/BX-XX

■ -XX: REF Surface / Epoxy color

Color \ Number	0	1	2	3	4
REF Surface Color	<input type="radio"/> White	<input type="radio"/> Black	<input type="radio"/> Gray	<input type="radio"/> Red	<input type="radio"/> Green
Epoxy Color	<input type="radio"/> Water Clear	<input type="radio"/> White	<input type="radio"/> Red	<input type="radio"/> Green	<input type="radio"/> Yellow

## ■ Mechanical Dimensions



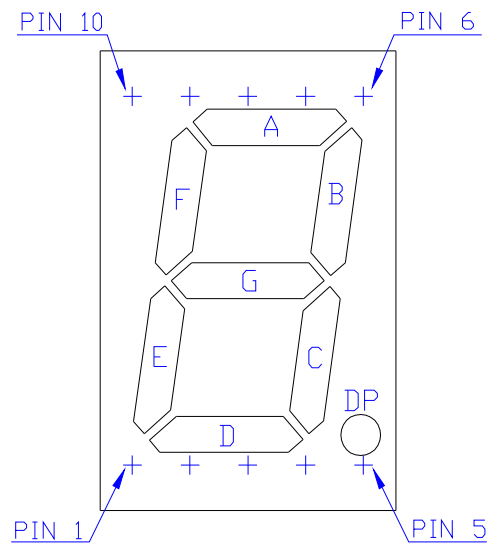
### Notes:

1. All pins are  $\Phi 0.51 [0.020]$  mm
2. Dimension in millimeter [inch], tolerance is  $\pm 0.25 [0.010]$  and angle is  $\pm 1^\circ$  unless otherwise noted.
3. Bending  $\leq$  Length \* 1%.
4. The specifications, characteristics and technical data described in the datasheet are subject to change without prior notice.



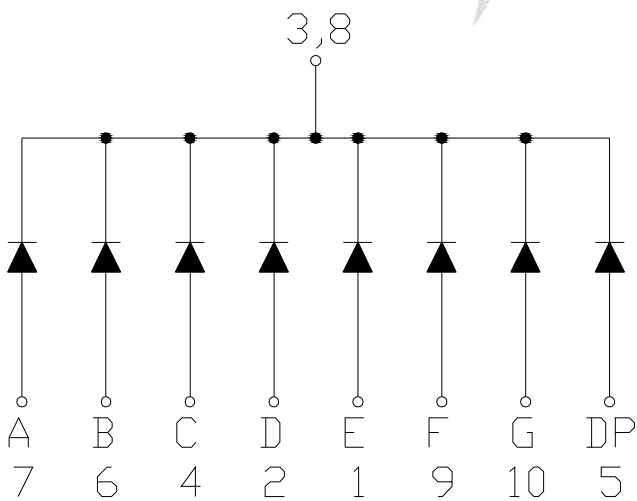
**Model No.: FYS-5612A/BX-XX**

■ **All Light On Segments Feature & Pin Position**



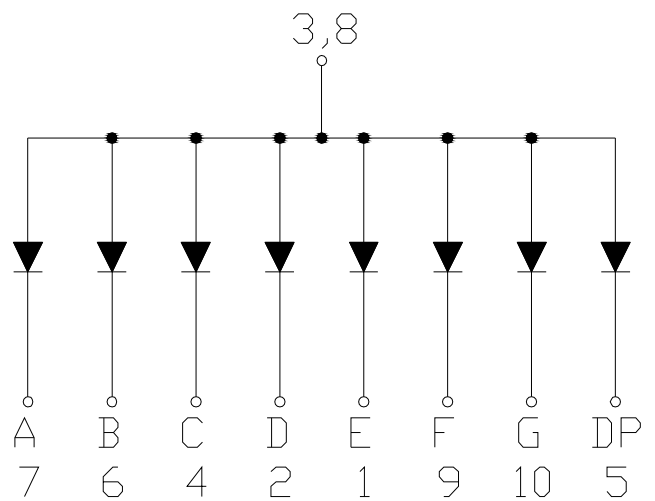
■ **Internal Circuit Diagrams**

FYS-5612AX



Common Cathode

FYS-5612BX



Common Anode

**Model No.: FYS-5612A/BX-XX**

■ **Absolute maximum ratings**

(Ta=25℃)

Parameter	Symbol	Test Condition	Value		Unit
			Min	Max	
Reverse Voltage	VR	IR=30	5	—	V
Forward Current	IF	—	—	30	mA
Power Dissipation	Pd	—	—	100	mW
Pulse Current	Ipeak	Duty=0.1mS,1KHz	—	150	mA
Operating Temperature	Topr	—	-40	+85	℃
Storage Temperature	Tstr	—	-40	+85	℃

■ **Electrical-Optical Characteristics**

● Color Code & Chip Characteristics:(Test Condition:IF=20mA)

(Ta=25℃)

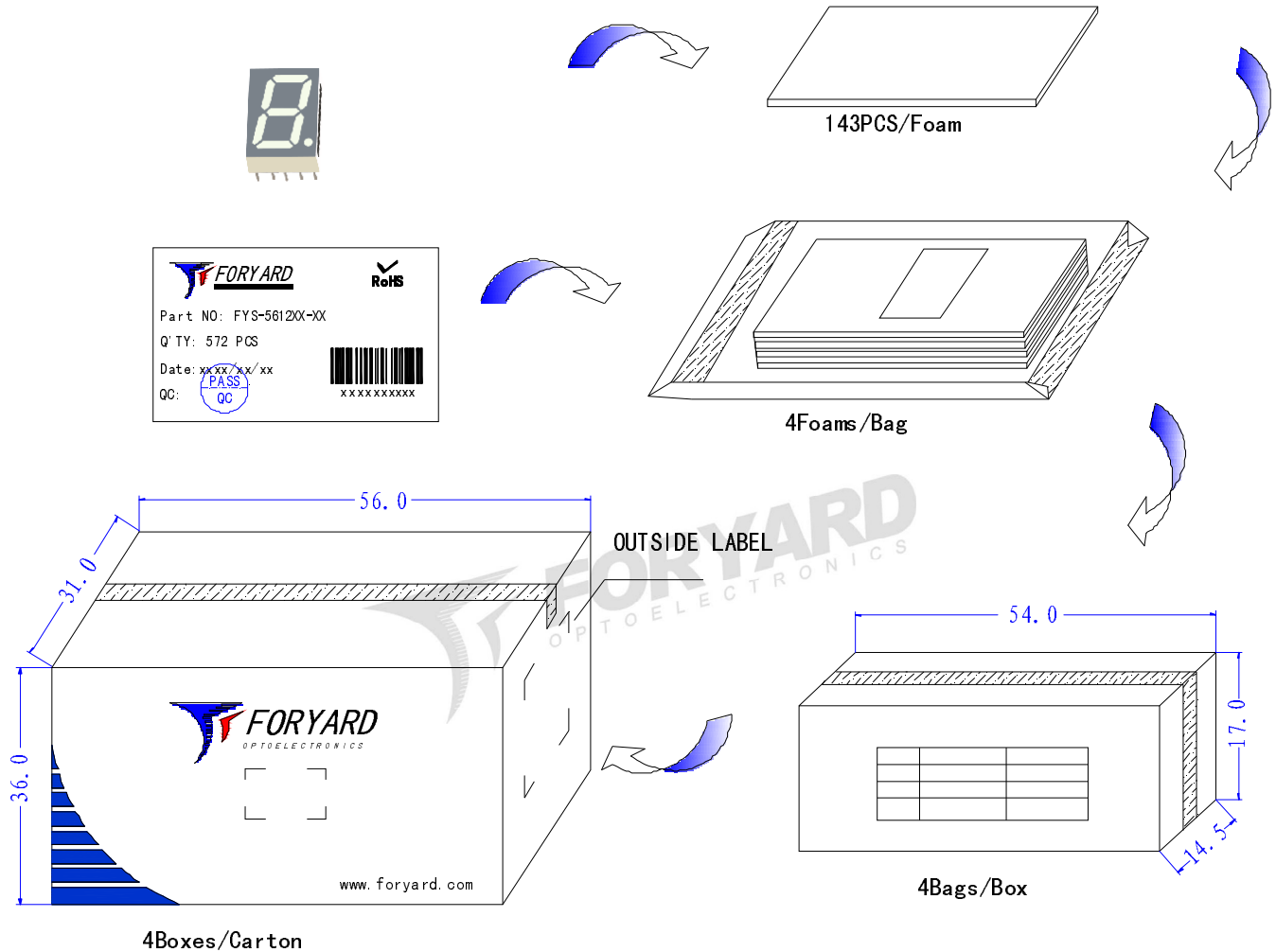
Emitting Color		Dice Material	Peak Wave Length( $\lambda_p$ )	Spectral Line halfwidth( $\Delta\lambda_{1/2}$ )	Forward Voltage(VF) Unit:V		Luminous Intensity (Iv) Unit:mcd
					Typ	Max	
Standard brightness							
H	Red	GaP	700nm	90nm	2.00	2.50	1
S	Hi Red	AlGaAs/SH	660nm	20nm	1.80	2.50	15~20
D	Super Red	AlGaAs/DH	650nm	20nm	1.90	2.50	26~38
E	Orange	GaAsP	625nm	35nm	1.90	2.50	14~20
A	Amber	GaAsP	610nm	35nm	1.90	2.50	13~18
Y	Yellow	GaAsP	590nm	35nm	1.90	2.50	13~18
G	Green	GaP	570nm	10nm	1.90	2.50	14~18
B	Blue	InGaN	430nm	60nm	3.40	4.40	0.7~1(mw)
			460nm		2.80	3.80	6~12(mw)
			470nm		2.80	3.80	6~12(mw)
PG	Pure Green	InGaN	520nm	36nm	2.80	3.80	4~6(mw)
W	White	InGaN	X=0.29,Y=0.30	CCT:9500K	2.80	3.80	20~30
Ultra brightness							
UHR	Ultra Hi Red	AlGaInP	640nm	20nm	1.90	2.50	30~60
UR	Ultra Red	AlGaInP	635nm	20nm	1.90	2.50	60~100
UE	Ultra Orange	AlGaInP	625nm	20nm	1.90	2.50	60~120
UA	Ultra Amber	AlGaInP	610nm	20nm	1.90	2.50	40~100~150
UY	Ultra Yellow	AlGaInP	590nm	20nm	1.90	2.50	50~140~190
UG	Ultra Green	AlGaInP	570nm	30nm	1.90	2.50	30~60~80
PG	Ultra Pure Green	InGaN	520nm	36nm	2.80	3.80	260~310
BG	Ultra Bluish Green	InGaN	505nm	36nm	2.80	3.80	260~310
UB	Ultra Blue	InGaN	460nm	30nm	2.80	3.80	80~90~120
			470nm	30nm	2.80	3.80	80~90~120
UW	Ultra White	InGaN	X=0.29,Y=0.30	CCT:9500K	2.80	3.80	180~200
Segment-to-Segment Luminous Intensity ratio(Iv-M)				1.5:1			

Note:

- 1.Luminous Intensity is based on the Foryard standards.
- 2.Pay attention about static for InGaN

**Model No.: FYS-5612A/BX-XX**

## ■ Packing Diagram



OUTSIDE LABEL

Note: The specifications are subject to change without notice. Please contact us for updated information.