به نام خدا

گزارش پروژه نهایی ریز پردازنده

طراحی و پیاده‌سازی Oscilloscope

محمدمهدی خانی

97243027

تیر 99

مقدمه

در این پروژه با استفاده از Keil و با زبان C، در محیط Proteus عملکرد یک Oscilloscope را شبیه سازی کرده ایم. پروژه روی نسخه Keil 5.34 و Proteus 8.10 تست شده است. بدلیل سنگین بودن برنامه، اجرای شبیه سازی در پروتئوس کند است.

اجزاء بکار رفته در پروژه

Sampling Unit

- STM32F401RE
- Keypad
- LCD

Display Unit

- STM32F401RE
- GLCD

روند کار کلی سیستم

بخش نمونه گیری

- در بخش Sampling Unit یک سیگنال آنالوگ به میکروکنترلر مبدا وصل شده است
- میکروکنترلر مبدا سیگنال آنالوگ را به مقادیر دیجیتال تبدیل می‌کند
- مقادیر دیجیتال با استفاده از ارتباط سریال به میکروکنترلر مقصد فرستاده می‌شوند

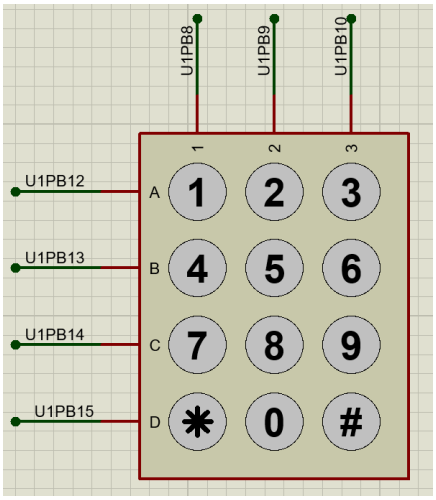
بخش نمایش

- میکروکنترلر مقصد مقادیر ارسالی توسط میکروکنترلر مبدا را دریافت می‌کند
- میکروکنترلر مقصد بر اساس مقادیر دریافتی و با استفاده از تابع داده شده مقدار مورد نظر را محاسبه میکند و مقادیر محاسبه شده را روی GLCD نمایش میدهد

کارکرد هر کدام از اجزاء سیستم

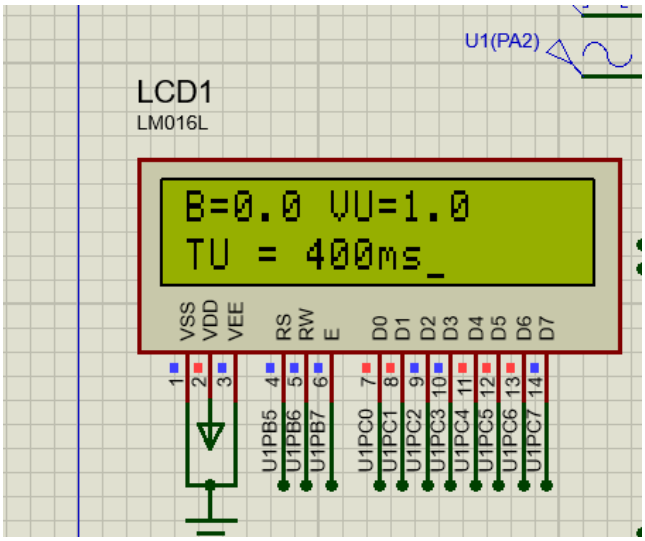
Keypad

برای خواندن مقدار فشرده شده همانطور که در تمرین های 6 و 7 عمل کردیم، در صفحه کلید ستون ها را به عنوان ورودی و سطر ها را به عنوان خروجی های STM32 در نظر میگیریم. سطر ها را یکی یکی و به نوبت فعال میکنیم و با خواندن ستون ها متوجه میشویم که کدام کلید فشرده شده است. تا زمانی که کاربر کلید را نگه داشته است Busy wait می‌کنیم (**Debounce button click**) و منتظر می‌مانیم تا کاربر کلید را رها کند. با استفاده از Keypad مقادیر A و B و Time unit و Voltage unit را کم و زیاد میکنیم.



LCD

مقادیر B و Time unit و Voltage unit را روی LCD نمایش میدهیم. کلاک پورت B را فعال میکنیم، پین های PTB4 و PTB5 و PTB6 را برای کانفیگ کردن این نمایشگر و کنترل رجیستر های LCD استفاده میکنیم. نمایشگر در Mode 8 بیتی کار میکند و پین های پورت C وظیفه انتقال داده به این نمایشگر را دارند. از LCD چیزی نمیخوانیم برای همین RW را همیشه روی W ست میکنیم. با هر لبه ی بالارونده E، مقادیر پین های C در LCD ذخیره میشوند.



ADC

پین های 1 و 2 از پورت A از میکروکنترلر مقصد به عنوان کانال های ورودی برای سیگنال آنالوگ تعریف شده اند. در تابع PortsInit کلاک های پورت A و ADC را فعال میکنیم. Resolution در حالت 12 بیتی ست شده است و برای تبدیل و ارسال 12 بیت یه داده 8 بیتی از شیفت منطقی استفاده شده است. نمونه برداری از سیگنال های آنالوگ توسط TIM5 مدیریت میشود و هر 20 میکروثانیه یکبار انجام میشود (نمونه برداری از 2 کانال).

اینتراپت مربوط به TIM5

هر 20 میکرو ثانیه یکبار TIM5 یک INT میدهد و این INT باعث میشود کنترل اجرای برنامه از Main گرفته شود و ISR مربوط به INT مورد نظر فرخوانی شود که در این Interrupt service routine از کانال های 1 و 2 نمونه برداری میشود (128 بایت).

اینتراپت مربوط به TIM2

هر 400 میلی ثانیه یکبار TIM2 یک INT میدهد و این INT باعث میشود کنترل اجرای برنامه از Main گرفته شود و ISR مربوط به INT مورد نظر فرخوانی شود که در این Interrupt service routine از نمونه هایی که قبلا از کانال های 1 و 2 نمونه برداری شده بود و در Buffer ذخیره شده بود از طریق ارتباط سریال به میکروکنترلر مقصد ارسال میشوند.

ارتباط سریال

برای ارتباط سریال بین دو میکروکنترلر از USART1 استفاده کرده ایم و کلاک مربوط به پورت A و USART1 فعال شده. کاربری پین های PTA9 و PTA10 را به عنوان فرستنده (Transmitter) و گیرنده (Receiver) ست کرده ایم و دو میکروکنترلر را به صورت ضربدری (Cross) به هم متصل کرده ایم (Transmitter A → Receiver B و برعکس). در هر بار ارسال داده، پکیج ارسالی شامل 8 بیت Data و 1 بیت تحت عنوان Stop

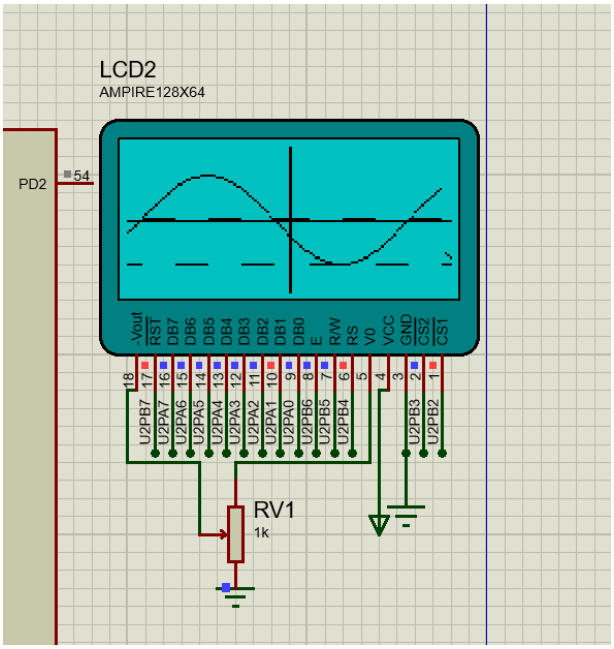
bit و و فاقد Parity bit است. در میکروکنترلر مبدا تابع ارسال سریال و در میکروکنترلر مقصد تابع دریافت سریال پیاده سازی شده اند.

محاسبه مقدار Y از روی X دریافتی

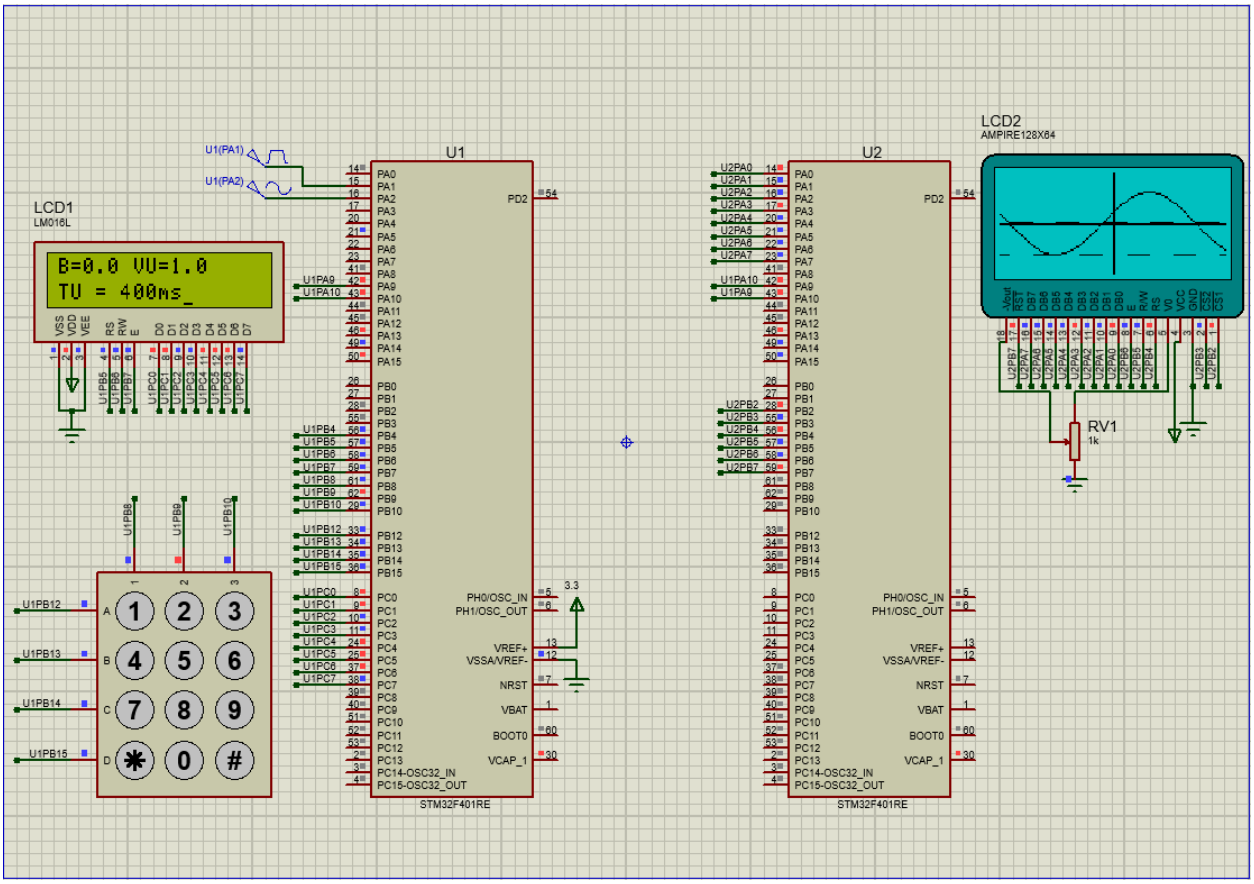
مقادیر دریافتی در میکروکنترلر مقصد مقادیر 8 بیتی هستند و بازه اعداد 0 تا 255 را در بر میگیرند. با توجه به اینکه ارتفاع GLCD بکار رفته 64 پیکسل است (8 لاین و هر لاین 8 پیکسل) این مقادیر را به این بازه Map میکنیم (با استفاده از شیفت). مقادیر بدست آمده را با توجه به مقادیر ست شده برای A و B در معادله AX + B قرار میدهیم.

نمایش مقدار Y روی GLCD

با توجه به اینکه ارتفاع GLCD 64 پیکسل است مقادیر Y اگر بیشتر از 63 باشند روی GLCD نمایش داده نمیشوند (اینگونه فرض میشود که در خارج از GLCD نمایش داده شده اند). اگر مقدار Y در محدوده 0 تا 63 بود برای روشن کردن پیکسل مورد نظر صفحه مناسب (CS1 یا CS2) را با توجه به ترتیب داده های دریافتی انتخاب میکنیم. در هر صفحه 8 لاین وجود دارد و هر لاین 64 ستون دارد و **هر ستون از هر لاین** خود به 8 پیکسل تقسیم میشود. برای دسترسی به پیکسل مورد نظر و روشن یا خاموش کردن آن باید صفحه مربوط به آن پیکسل، لاین مربوط به آن پیکسل و ستون مربوط به آن پیکسل مشخص شوند، سپس با یک بایت (هر بیت از این بایت متناظر با یک پیکسل از آن 8 پیکسل است) این پیکسل ها را کنترل میکنیم.

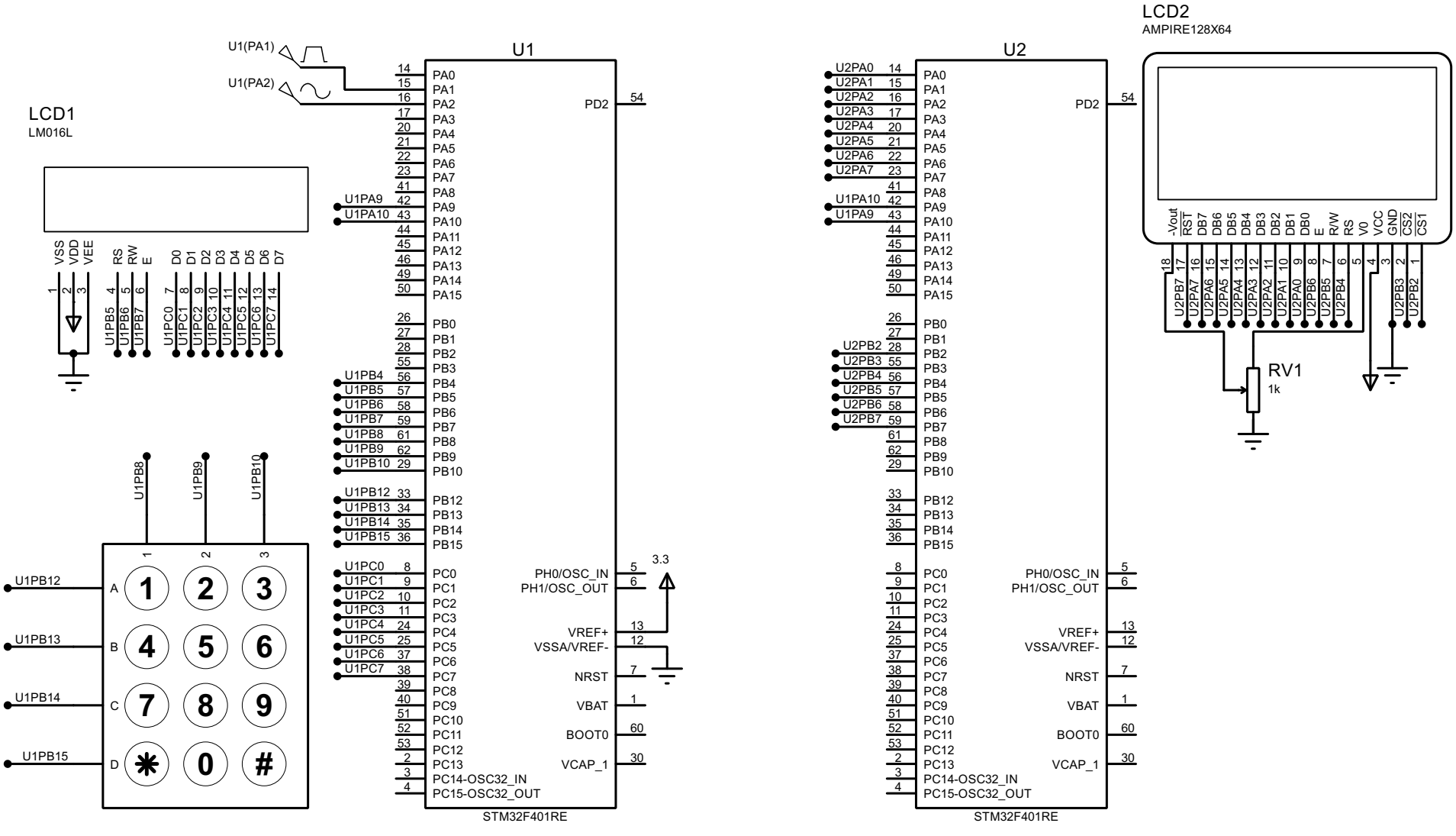


تصویر از محیط اجرا



خروجی گرافیکی Proteus

خروجی پروتئوس در صفحه بعد آورده شده است.


LCD1
LM016L
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
U1
U1(PA1)
U1(PA2)
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB12
PB13
PB14
PB15
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
PC11
PC12
PC13
PC14-OSC32_IN
PC15-OSC32_OUT
PD2
PH0/OSC_IN
PH1/OSC_OUT
VREF+
VSSA/VREF-
NRST
VBAT
BOOT0
VCAP_1
3.3
STM32F401RE
U1PB8
U1PB9
U1PB10
U1PB12
U1PB13
U1PB14
U1PB15
A
B
C
D
1
2
3
4
5
6
7
8
9
*
0
#
U2
STM32F401RE
LCD2
AMPIRE128X64
-Vout
RST
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
E
R/W
RS
V0
VCC
GND
CS2
CS1
RV1
1k