

دوره آموزش پیشرفته STM32F103

elcen.ir

اگر اهداف زیر رو دارید با این دوره ی آموزشی سریعتر و راحت تر به هدفتون میرسید.

- خودت بتونی با میکروکنترلر STM32 پروژه انجام بدی.
- استفاده از امکانات میکروکنترلر STM32F103 برای حرفه ای تر انجام دادن پروژه ها.
- خودت کتابخانه راه اندازی قطعات رو از صفر بنویسی.
- هم کتابخانه HAL و هم برنامه نویسی رجیستری رو مسلط بشی.

ویژگی های دوره

پروژه محور:

در جریان راه اندازی قطعات ، استفاده از امکانات میکروکنترلر رو یادمیگیری.

استفاده از امکانات STM32 :

میکروکنترلرهای STM32 قدرت هایی دارند که میکروکنترلرهای ارزان تر ندارند، در این دوره انجام پروژه ها رو با استفاده از قابلیت های قدرتمند STM32 و با روش هایی که با آردوینو و AVR امکانش وجود نداره، یادمیگیرید.

برنامه نویسی حرفه ای :

مثلا در پروژه ها از تابع تاخیر یا delay استفاده نمیکنیم و هر جا ممکن باشه از DMA استفاده میکنیم و تا جای ممکن عملیات ها رو پریفرال ها انجام میدن به جای اینکه CPU کار کنه.

شکل دوره

در هر جلسه دیتاشیت و نحوه ی عملکرد یک قطعه بررسی میشه، در قدم بعد کتابخونه راه اندازی با میکروکنترلر STM32F103 با روش برنامه نویسی رجیستری از صفر نوشته میشه و کاربرد هر بیت رجیسترها بررسی میشه. همچنین توابع پروژه رو با کتابخونه HAL هم انجام میدیم و توابع کتابخونه HAL بررسی میشه.

روش مشاهده دوره

باید همزمان با دوره خودتون هم پروژه ها رو انجام بدید. حتما وقتی رو برای مشاهده ی دوره و انجام تمرین ها در نظر بگیرید.

فصل اول : تایمر و DMA

معرفی تایمرها

- کاربردهای پریفرال تایمر میکروکنترلرهای stm32
- TIMBASE و رجیسترهای شمارنده و ARR و PSC
- محاسبه ی ARR و PSC با تنظیم دوره تناوب کانتر و دوره تناوب کلاک کانتر
- رجیسترهای CCR و کانال در حالت ورودی و خروجی
- کانتر بالارونده ، پایین رونده و edge aligned ، خروجی PWM1 و PWM2
- قابلیت پرلود و رجیستر سایه
- حالت تک پالس
- کنترل نرخ update event با رجیستر RCR
- حالت تک پالس با رجیستر RCR و تولید تعداد مشخص پالس
- انواع تایمرها در STM32

اندازه گیری سرعت اجرای برنامه در میکروکنترلر STM32 با سه روش

- پروژه led چشمک زن با استفاده از اینتراپت آپدیت تایمر در STM32
- ARR و PSC برای دقیق ترین تایمر ممکن در stm32f103c8
- اندازه گیری سرعت اجرای کد STM32 با یک تایمر، توابع شروع و توقف شمارنده و خوندن CNT
- معرفی و بررسی رجیسترهای راه اندازی شمارنده ۳۲ بیتی CYCCNT از واحد DWT
- اندازه گیری سرعت اجرای کد با شمارنده داخل CPU cortex-m3
- افزایش ظرفیت شمارنده با اتصال دو تایمر در حالت master-slave ، تنظیم تریگرهای ورودی و خروجی
- اندازه گیری سرعت اجرای کد با دو تایمر آبشاری در میکروکنترلر STM32

پین ETR و پالس شمار سخت افزاری با تایمر STM32

- پالس شمار نرم افزاری : شمارش و مقایسه ی تعداد پالس ها توسط CPU در اینتراپت خارجی

- پالس شمار سخت افزاری : کاربرد محاسبه تعداد دفعات عبور جسم از مقابل سنسور proximity
- شمارش پالس ها با قابلیت external clock و پین ETR تایمر STM32
- مقایسه ی تعداد پالس ها و تولید اینتراپت توسط کانال خروجی در حالت frozen

فاصله سنج التراسونیک با STM32 بدون دیلی | ارسال و دریافت پالس با تایمر

- دیتاشیت سنسور : SR-04 پینها، پالس تریگ و پالس اکو، زمان بندی
- تفاوت راه اندازی sr04 در stm32 و AVR ، استفاده از تایمر برای ارسال و دریافت پالس
- SR-04 با stm32f103 بدون دیلی :
- ساخت پالس تریگ با کانال خروجی تایمر STM32 در حالت PWM
- اندازه گیری مدت زمان یک بودن پالس echo با دو کانال ورودی input capture

lcd کاراکتری با STM32 بدون delay با تایمر و DMA | پیاده سازی سخت افزاری ارتباط پارالل

- دیتاشیت lcd کاراکتری : پین ها، چگونگی نمایش کاراکتر، حافظه ها ROM و RAM ، کامند ها، عملیات استارت lcd.
- کنترل زمان بندی بستر سخت افزاری ارتباط پارالل با تایمر stm32 و انتقال داده به باس پارالل با DMA میکروکنترلر stm32f103.
- کانال تایمر STM32 در حالت خروجی PWM برای پین E ارتباط پارالل
- تغییر وضعیت پین های باس پارالل در اینتراپت کانال خروجی تایمر میکروکنترلر STM32
- DMA به : GPIO تغییر وضعیت پین های باس پارالل توسط DMA با نوشتن در رجیستر BSRR
- آماده سازی داده ها برای نوشته شدن در رجیستر BSRR برای مد ۸ بیت و ۴ بیت ارتباط با lcd کاراکتری
- ارسال تعداد مشخص بایت با استفاده از قابلیت تک پالس و رجیستر RCR تایمر های STM32

سنسور دما و رطوبت DHT22 با STM32 بدون دیلی با تایمر و DMA | پیاده سازی سخت افزاری ارتباط ۱-wire-

- دیتاشیت سنسور دما و رطوبت : DHT22 پین ها، پروتکل ارتباطی
- چگونگی پیاده سازی بستر سخت افزاری ارتباط one wire توسط تایمر stm32f103
- کانال خروجی تایمر STM32 در حالت PWM برای ارسال پالس استارت و زمین کردن باس
- یک کردن خروجی PWM بعد از توقف تایمر
- دو کانال ورودی تایمر STM32 در حالت input capture برای اندازه گیری مدت زمان یک بودن ۴۰ پالس

- محاسبه ی ۴۰ بیت در ۴۰ اینتراپت لبه ی پایین رونده
- طراحی union برای نگهداری داده های سنسور
- تنظیم اینتراپت تایمر دوم برای استارت تایمر اصلی
- روش اول : استارت تایمر اول توسط تایمر دوم با قابلیت master/slave و بصورت سخت افزاری
- روش دوم : استفاده از کانال خروجی تایمر master برای زمین کردن باس one wire
- تنظیم دو کانال DMA برای انتقال اعداد رجیستر های CCR1 و CCR2 به دو آرایه ۴۳ عضوی
- روش سوم: انتقال اعداد از رجیسترهای CCR1 و CCR2 به حافظه توسط DMA و محاسبه ی بیت ها در یک اینتراپت TC

سنسور دما DS18B20 با STM32 بدون دیلی با تایمر و | DMA پیاده سازی سخت افزاری ارتباط ۱-wire-

- دیتاشیت سنسور دما : DS18B20 پینها، ترانزیستور C1815 برای زمین کردن باس one wire ، حافظه ها، بررسی transaction، کامند ROM و function،
- زمان بندی نوشتن و خواندن بیت های ۰ و ۱ از باس
- خواندن دما از DS18B20 با STM32 بوسیله ی دو تایمر و دو کانال DMA
- ارسال کامند ۸ بیتی با ارسال ۹ پالس PWM با کانال خروجی تایمر
- کانال DMA میکروکنترلر STM32 برای نوشتن ۹ مقدار عرض پالس PWM در رجیستر CCR3 و فرستادن یک بایت
- اندازه گیری عرض پالس ۷۲ بیت با کانال ورودی تایمر STM32 در حالت input capture
- کانال DMA برای انتقال ۷۲ مقدار رجیستر CCR1 و خواندن ۷۲ بیت
- طراحی union برای ذخیره سازی بیتها
- اینتراپت DMA برای محاسبه ی بیت ها و CRC ، کتابخانه ی CRC
- نوشتن تابع برای انجام ۷ عملیات با فاصله ها زمانی نامنظم با استفاده از قابلیت پریلود رجیستر ARR تایمر STM32F103

انکودر با تایمر STM32

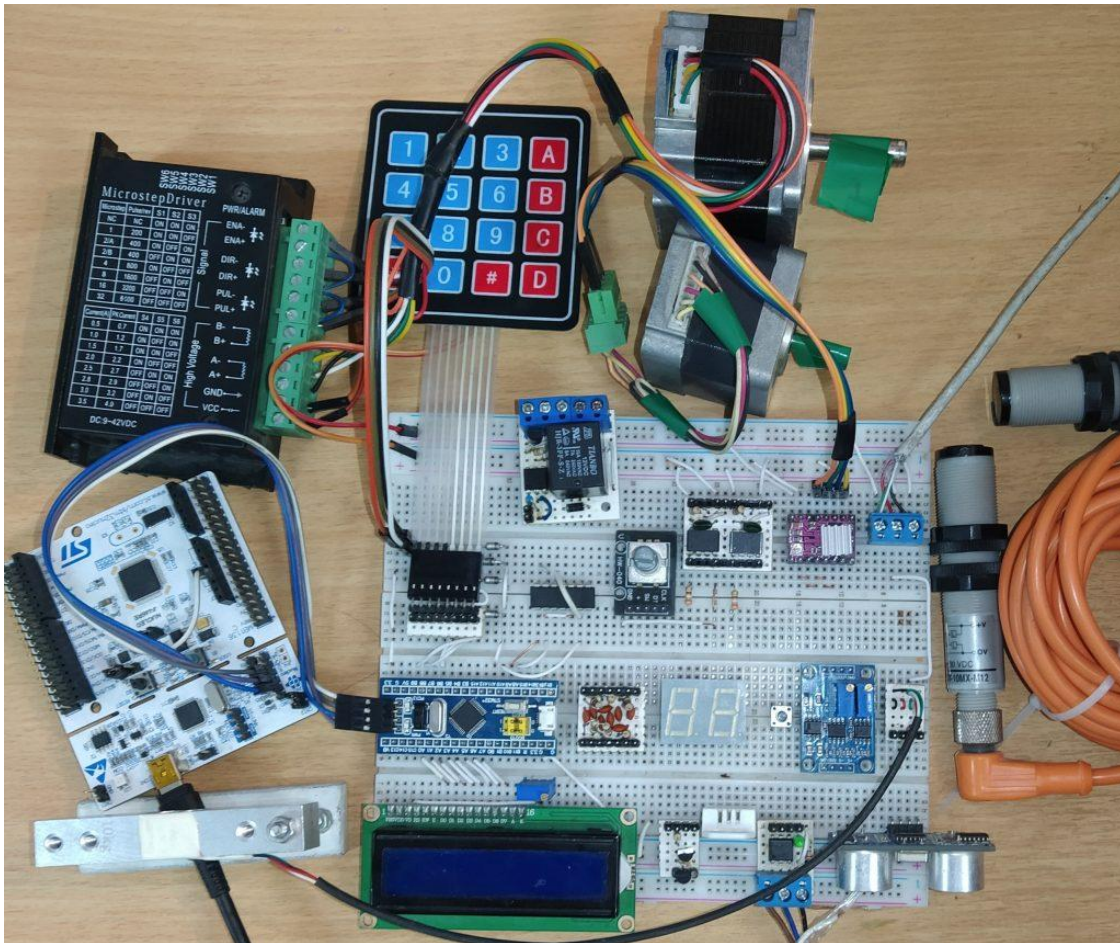
- خروجی انکودر incremental ، روتاری انکودر، تشخیص سرعت و جهت با شمردن لبه با توجه به وضعیت کانال دیگر
- ورودی های TI1FP1 و TI2FP2 ، بررسی تفاوت سه حالت مختلف شمارش پالس های انکودر توسط تایمر STM32
- شمارش پالس های روتاری انکودر و تشخیص جهت با تایمر STM32

ورودی و خروجی PWM با تایمر STM32

- فیلتر پایین گذر خازن مقاومتی ، فرکانس شکست
- پروژه خروجی آنالوگ با PWM در میکروکنترلر STM32
- اندازه گیری فرکانس و مدت زمان یک بودن PWM ورودی با دو کانال ، TI1FP1تریگر ورودی

کنترل سرعت و موقعیت استپر موتور با میکروکنترلر | STM32 ارسال پالس با تعداد و فرکانس مشخص

- درایور : DRV8825 تغذیه ها، پین های dir, step, sleep, reset, enable مشخص کردن رزولوشن و محدود کردن دما با پتانسیومتر
- مدار ایزوله کردن پین های STEP و DIR با دو اپتوکوپلر N137۶
- درایور : TB6600 تغذیه و اتصال به موتور و میکروکنترلر ، محدود کردن جریان و مشخص کردن رزولوشن با DIP switch
- دیتاشیت استپر موتور
- رابطه ی سرعت زاویه ای با فرکانس PWM خروجی و مقدار رجیسترهای ARR و PSC
- مقایسه ی رزولوشن تولید سرعت زاویه ای در بازه های مختلف سرعت برای دو حالت $ARR=1$ و $PSC=0$
- کانال یک در حالت pwm برای تولید پالس step
- پروژه تنظیم سرعت و جهت استپر موتور با درایور ها DRV8825 و : TB6600 خروجی PWM با فرکانس متغیر
- کنترل موقعیت استپر موتور : مشخص کردن تعداد پالس های خروجی TIM3 با TIM1 با master/slave دو طرفه
- استفاده از TIM1 برای شمارش آپدیت های TIM3 و توقف TIM3 با قابلیت master/slave



فصل دوم : پریفرال ADC در میکروکنترلر STM32F1

نمونه برداری از تک کانال

- بررسی عملکرد پریفرال ADC در میکروکنترلر STM32F1 ، مبدل ADC ، کانال های ADC
- گروه regular و injected ، دیتا رجیسترهای DR و JDR ، صف کانال ها در رجیسترهای SQR و JSQR در ADC میکروکنترلر STM32F1
- تنظیم زمان نمونه برداری و بیت های SMP در ADC
- پروژه اول : حالت کاری تک کانال ادامه دار و بررسی کاربردها ، single channel continuous conversion
- پروژه دوم : حالت کاری تک کانال تک تبدیل و بررسی کاربردها ، single channel single conversion

- پروژه ی سوم : تنظیم تریگر خارجی برای ADC ، زمان بندی تبدیل ADC با event تایمر ، بیت های EXTSEL ، EXTTRIG ، SWSTART
- نوشتن توابع IRQHandler برای اینترایت های ADC و مدیریت اینترایت
- پروژه چهارم : تعیین وضعیت خروجی دیجیتال با توجه به مقدار ADC با استفاده از اینترایت ADC
- معرفی analog watchdog در میکروکنترلر STM32F1 و بیت های AWDCH و AWDEN و AWDSGH
- پروژه پنجم : مقایسه مقدار خروجی ADC با استفاده از watchdog event
- پروژه ششم : خواندن دما از سنسور داخلی میکروکنترلر با کانال ۱۶ و محاسبات دما ، استفاده از analog watchdog روی سنسور دمای داخلی
- پروژه هفتم : استفاده از DMA برای انتقال داده های ADC به حافظه
- نکته ADC : هنوز چند جلسه داره تا تکمیل شه

فصل سوم : پریفرال RTC در میکروکنترلر | STM32F1 ساعت ، تقویم و Alarm

اینترپت ثانیه با RTC

- تنظیمات اولیه پریفرال RTC و اینترپت ثانیه
- تنظیمات و فعالسازی کریستال خارجی LSI ، بیت های DBP ، LSEON ، RTCSEL ، LSERDY و RTCEN
- ساختار پریفرال RTC در STM32F1 ، معرفی رجیسترهای ALR ، DIV ، PRL ، CNT و CR ، معرفی Backup domain و APB1 domain
- بیت RTOFF و نوشتن در رجیسترهای پریفرال RTC ، مکانیزم چک کردن همراه با timeout
- نوشتن توابع برای ورود و خروج از حالت تنظیمات پریفرال RTC ، بیت CNF
- بیت RSF و مکانیزم چک کردن سنکرون بودن RTC
- نوشتن توابع IRQHandler و مدیریت اینترپت های RTC
- پروژه ساخت اینترپت ثانیه با پریفرال RTC

تنظیم ساعت RTC و نمایش ساعت در LCD ، آپدیت تاریخ در اینترپت Overflow

- نوشتن تابع دریافت ساعت از labview با USART
- خواندن رجیستر CNT در پریفرال RTC و محاسبه ساعت ، کاربرد یونیون برای ذخیره سازی مقدار CNT
- جداسازی تنظیمات اولیه و ثانویه RTC ، بردن و خارج کردن میکروکنترلر به حالت تنظیم اولیه RTC با پین GPIO
- نمایش ساعت در lcd کاراکتری در اینترپت ثانیه RTC
- تست استفاده از باتری برای تامین برق بخش backup domain و پریفرال RTC
- تابع دریافت تقویم از labview با USART بصورت رجیستری
- آپدیت کردن تاریخ در انتهای هر روز در اینترپت Overflow در پریفرال RTC میکروکنترلر STM32F1

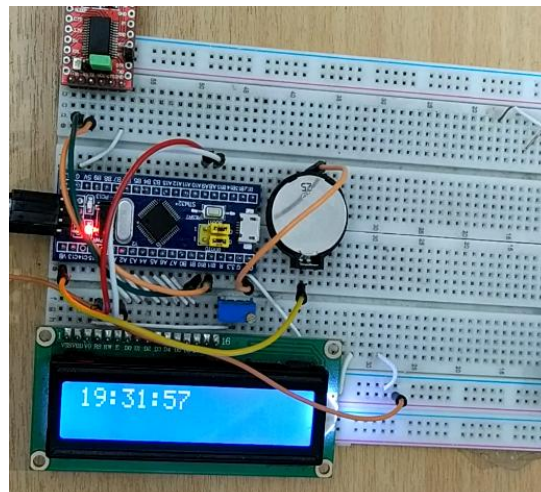
پریفرال RTC و کتابخانه HAL

- تنظیمات LSI و RTC در نرم افزار STM32CubeIDE

- بررسی استراکچر RTC_HandleTypeDef
- بررسی تابع HAL_RTC_Init و HAL_RTC_MspInit
- فرمت نمایش BCD و Binary و تبدیل ها
- بررسی warning های کتابخانه HAL برای تنظیمات RTC در STM32F1
- فعال کردن اینتراپت ثانیه RTC با کتابخانه HAL و پروژه اینتراپت ثانیه
- بررسی تابع HAL_RTCEX_RTCIRQHandler
- تابع HAL_RTC_SetTime بررسی
- جداسازی تنظیمات اولیه و ثانویه RTC در کتابخانه HAL ، دریافت زمان از labview با USART و تنظیم ساعت
- روش آپدیت تاریخ در کتابخانه HAL و بررسی تابع RTC_DateUpdate
- رفع مشکل از دست رفتن تاریخ در کتابخانه HAL ، معرفی رجیسترهای Backup و ذخیره تاریخ در رجیسترهای Backup
- نوشتن کد labview برای ارسال تاریخ به میکروکنترلر توسط USART
- نوشتن کد دریافت تاریخ در میکروکنترلر از labview و تنظیم تاریخ

Alarm در RTC میکروکنترلر STM32F1 رجیستری

- مدیریت و فعالسازی اینتراپت Alarm در EXTI17
- نوشتن تابع IRQHandler برای اینتراپت Alarm
- بررسی سیگنال RTC_Alarm و تفاوت استفاده از لبه بالا رونده و لبه ی پایین رونده سیگنال RTC_Alarm
- مدیریت تعداد دلخواه Alarm در هر روز ، تشخیص Alarm با خواندن رجیستر CNT
- انجام پروژه تنظیم سه Alarm همراه با اینتراپت



فصل چهارم : پریفرال PWR و حالت های کم مصرف در میکروکنترلر STM32F1

اخطار افت ولتاژ

- اخطار افت ولتاژ PVD در میکروکنترلر STM32F1
- voltage domain ها در میکروکنترلر STM32F103 ، بازه ها و محدودیت ها
- معرفی POR و PDR
- معرفی PVD و سیگنال PVD output ، تنظیم حد ولتاژ با بیت های PLS ، فعالسازی با بیت PVDE
- نوشتن تابع IRQHandler برای اینتراپت PVD
- فعالسازی اینتراپت PVD در EXTI16
- بررسی تفاوت اینتراپت PVD در لبه ی بالارونده و پایین رونده سیگنال PVD Output
- تشخیص لبه بالارونده از پایین رونده با بیت PVDO
- پروژه اخطار در صورت افت ولتاژ در میکروکنترلر STM32F1

مدیریت مصرف توان در میکروکنترلر STM32F1 و حالت کم مصرف sleep

- بررسی مثال های مدیریت مصرف توان در حالت عادی با پایین آوردن کلاک و خاموش کردن کلاک پریفرال ها
- بررسی حالت های کم مصرف CPU و رگولاتور و بیت های SLEEPDEEP و PDDS
- تفاوت حالت های کم مصرف sleep ، stop و standby و تفاوت روشهای sleep on exit ، WFI و WFE برای ورود به حالت های کم مصرف
- وضعیت کلاک HCLK و FCLK و کلاک SysTick در حالت های کاری کم مصرف و غیرفعال کردن SysTick
- پروژه اول: حالت کاری کم مصرف sleep با : sleep on exit فقط اجرای روتین اینتراپت ها
- فعال کردن قابلیت دیباگ در حالت کم مصرف sleep با بیت DBG_SLEEP
- مروری بر inline assembly ، نکات استفاده از WFI ، استفاده از ISB و DSB
- روش های مختلف لود کردن کد در حالت های کم مصرف: دکمه ریست ، استفاده از بوت لودر ، استفاده از st-link utility
- پروژه دوم: حالت کاری کم مصرف sleep با WFI در main
- بررسی عمکرد WFE و SEV
- پروژه سوم: حالت کاری کم مصرف sleep با WFE ، تولید ایونت با اینتراپت EXTI ، با ایونت EXTI و با ایونت تایمر

- روش فعالسازی ایونت برای پریفرال ها بجز EXTI و عملکرد بیت SEVONPEND

حالت کم مصرف stop در میکروکنترلر STM32F1

- تعیین وضعیت بیت های SLEEPDEEP و PDDS و LPDS برای حالت stop
- نوشتن تابع تنظیمات مجدد کلاک بعد از خروج از حالت stop
- پروژه اول : رفتن به حالت stop با sleep-on-exit
- پروژه دوم : رفتن به حالت stop با WFI ، فراخوانی WFI در main و در اینتراپت ، بررسی اولیت اینتراپت برای فراخوانی WFI
- پروژه سوم: خروج از حالت stop با استفاده از ایونت پریفرال EXTI
- فعالسازی دیباگ در حالت stop با بیت DBG_STOP
- پروژه چهارم: خروج از حالت کم مصرف stop در ساعت مشخص با RTC Alarm

حالت کم مصرف standby در میکروکنترلر STM32F1

- فلگ WUF و صفر کردن آن با بیت CWUF
- فلگ SBF برای تشخیص حالت standby و صفر کردن با CSBF
- فعال کردن پین PA0 برای خروج از حالت standby با بیت EWUP و بررسی مدار مورد نیاز و بررسی اشکالات power on
- پروژه اول : خروج از حالت standby با استفاده از پین wake up
- خارج کردن اینتراپت SysTick از وضعیت pending با رجیستر ICSR
- پروژه دوم: رفتن به حالت Standby در روتین اینتراپت
- پروژه سوم: خروج از حالت Standby در ساعت مشخص شده با RTC Alarm

فصل پنجم: پریفرال USART در میکروکنترلر STM32F1 و ارتباط با کامپیوتر

- معرفی ارتباط سنکرون و آسنکرون، معرفی پین های USART، معرفی single و half duplex و full duplex
- معرفی بستر سخت افزاری : USART طول کلمه، Baud rate، parity و stop bit
- فرستادن داده با USART به روش polling و با DMA در STM32 رجیستری
- بیت های UE، M، stop و TE، محاسبه مقدار بیت های Fraction و Mantissa با توجه به مقدار انتخابی baud rate
- بررسی دیتارجیستر USART و رجیسترهای ارسال و دریافت داده و شیفت رجیسترها
- فلگ TXE و فلگ TC برای اطلاع از انتقال داده توسط USART
- نوشتن تابع ارسال بایت ها با یوزارت به روش polling رجیستری
- پروژه printf با یوزارت به روش polling، پرینت پیغام ها در کامپیوتر در نرم افزار realterm
- تنظیم رجیستر CPAR و بیت های DIR، CIRC، PINC، MINC، PSIZE، MSIZE، PL برای انتقال داده ها با DMA به یوزارت
- نوشتن تابع استارت کردن عملیات DMA و رجیسترهای CMAP و CNDTR
- پروژه ی printf با یوزارت با DMA، پرینت پیغام ها در نرم افزار realterm
- بررسی کامل تمام ایونت ها و اینترپت های یوزارت: PE، FE، NE، ORE، IDLE، RXNE، TC، TXE، LBD، CTS :
- مدیریت اینترپت های یوزارت و نوشتن تابع : IRQ_Handler صفر کردن فلگ ها و فراخوانی تابع Callback

بررسی کتابخانه HAL برای پریفرال USART و مقایسه توابع polling, interrupt و DMA

- تنظیمات یوزارت در نرم افزار STM32CubeIDE
- تفاوت عملکرد سه روش انتقال دیتا به دیتا رجیستر یوزارت polling و interrupt و DMA
- بررسی عملکرد ورودی timeout در تابع HAL_UART_Transmit و اندازه گیری زمان اجرای این تابع، بررسی خروجی HAL_Timeout
- اندازه گیری زمان اجرای HAL_UART_Transmit_IT، بررسی عملکرد تابع با قابلیت watchpoint در دیباگ و خروجی HAL_Busy
- بررسی نقش اینترپت های DMA در کتابخانه HAL وقتی از DMA برای انتقال داده به یوزارت استفاده میشه
- نقش تابع MSP و بررسی ماکرو LINK_DMA برای اجرای اینترپت های DMA
- استفاده و اندازه گیری زمان اجرای تابع HAL_UART_Transmit_DMA و خروجی HAL_Busy
- اضافه کردن قابلیت timeout به کد رجیستری ارسال داده با یوزارت با استفاده از SysTick

- برنامه ی قابلیت تشخیص busy در کد رجیستری برای ارسال داده با یوزارت
- بررسی خروجی HAL_Error در توابع یوزارت

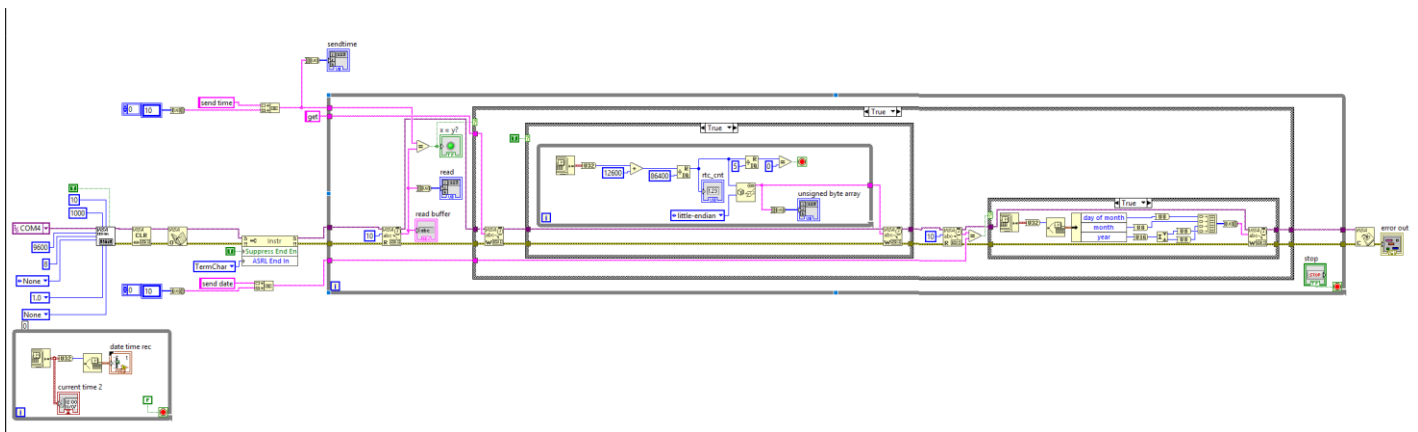
نوشتن برنامه های Labview برای ارتباط با میکروکنترلر STM32

- معرفی بخش های مختلف نرم افزار Labview ، نمایشگر و کنترلر
- بلوک visa config و انجام تنظیمات بستر سخت افزاری UART
- بلوک های open, read, clear, close در labview
- بررسی ۳ دلیل توقف بلوک read در labview و اخطار خروجی برای هر کدام
- شناسایی نوع اخطار های سریال در Labview با کلاستر اخطار ها در case
- مشخص کردن کاراکتر termination سریال با VISA property node
- بررسی مثال اخطار timeout در labview
- حلقه ی while در labview ، توقف و ایندکس حلقه
- چسباندن رشته ها به همدیگر و استفاده از شیفت رجیستر در labview
- پروژه printf با USART در Labview
- پروژه ی ارسال بایت به Labview از میکروکنترلر STM32

دریافت ساعت از Labview با استفاده از USART در میکروکنترلر STM32F1

- بلوک Get Date in second در Labview و بررسی زمان epoch در کامپیوتر
- محاسبه ی تعداد ثانیه های گذشته از ابتدای روز در Labview
- تبدیل عدد ۳۲ بیتی زمان به آرایه از چهار ۸ بیتی با بلوک های split number و کست کردن و build array
- بلوک flatten string و ارسال داده ها از Labview با بلوک send
- نقش فلگ RXNE در دریافت داده های USART در STM32
- استفاده و بررسی تابع HAL_UART_Recieve
- نوع union برای چهار بایت دریافتی زمان بوسیله ی USART
- سنکرون کردن ارتباط USART بین میکروکنترلر و Labview با استفاده از termination char
- اضافه کردن کاراکتر NULL به انتهای رشته در Labview
- تغییر کاراکتر termination char در labview

- نوشتن تابع مدیریت ارتباط USART برای دریافت زمان از labview همراه timeout
- تغییر تابع HAL_RTC_SetTime برای تنظیم RTC
- نوشتن تابع دریافت داده های USART بصورت رجیستری با روش polling
- پروژه دریافت زمان از labview با کتابخانه HAL و رجیستری



فصل ششم: ارتباط مدباس در Labview با PLC دلتا DVP14SS2

این فصل آموزش نرم افزار Labview است.

مدباس در Labview و ساخت HMI برای PLC دلتا DVP14SS2 در کامپیوتر

- معرفی پینها و سیم کشی PLC دلتا DVP14SS2
- نوشتن برنامه ی PLC برای Ascii و RTU در برنامه ی WPL Soft
- معرفی RS485 و مبدل های RS485 به USB و RS485 به TTL
- معرفی رجیسترها و رله ها در حافظه ی PLC
- بررسی ساختار دستورات مدباس و تفاوت مدباس Ascii و RTU
- روش محاسبه ی checksum در مدباس Ascii

نوشتن برنامه Labview برای تغییر خروجی های دیجیتال PLC با استفاده از Labview

- تنظیمات اولیه ارتباط سریال در Labview با بلوک configure port
- بلوک های open ، write ، read ، clear و close در labview
- بررسی کامند ۱۵ برای تغییر بیت ها و پیاده سازی در Labview
- معرفی آدرس ها در PLC
- ساخت دستور مدباس Ascii با بلوک concatenate string در Labview
- بلوک Build array و Boolean array to number در Labview
- تبدیل عدد به رشته ی hexadecimal در Labview و تبدیل رشته به عدد
- بلوک subsetstring و جدا کردن بخشی از یک رشته در Labview
- محاسبه ی LRC ارسالی و دریافتی در Labview
- بررسی پاسخ ارسالی از طرف PLC به کامپیوتر برای دستور 15
- تست برنامه Labview برای مدباس Ascii به روش اول
- بلوک format into string برای ساخت رشته
- استفاده از flat sequence در Labview برای اعمال ترتیب انجام کارها
- بلوک scan from string در Labview برای تفکیک داده های دریافتی از PLC

- بررسی و دریافت پیام های خطا که توسط PLC به Labview ارسال میشود
- تشخیص نوع پیام خطای ارسال شده توسط PLC در Labview با case structure
- بررسی اشتباه استفاده از local variable در Labview
- استفاده از کلاستر برای ذخیره سازی داده های دریافت شده از پنل
- بلوک های bundle by name و unbundle by name در Labview
- تست پروژه ی تغییر خروجی های دیجیتال PLC با Labview به روش دوم

نوشتن کتابخانه مدباس Ascii و RTU برای نوشتن و خواندن در کویل ها و رجیسترهای PLC DVP14SS2

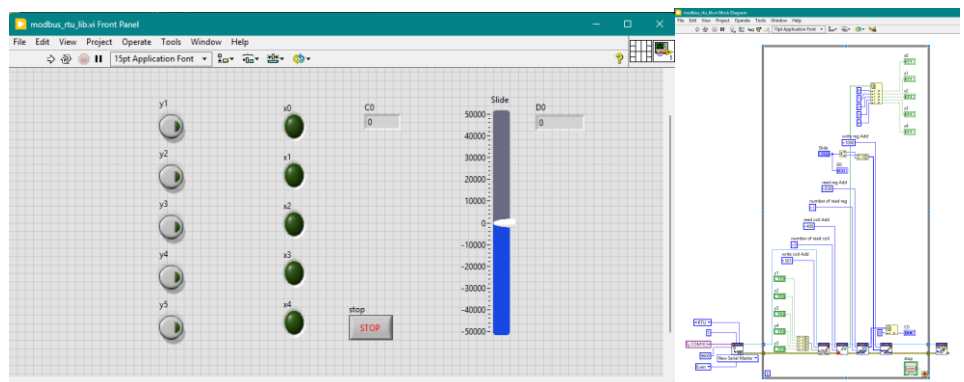
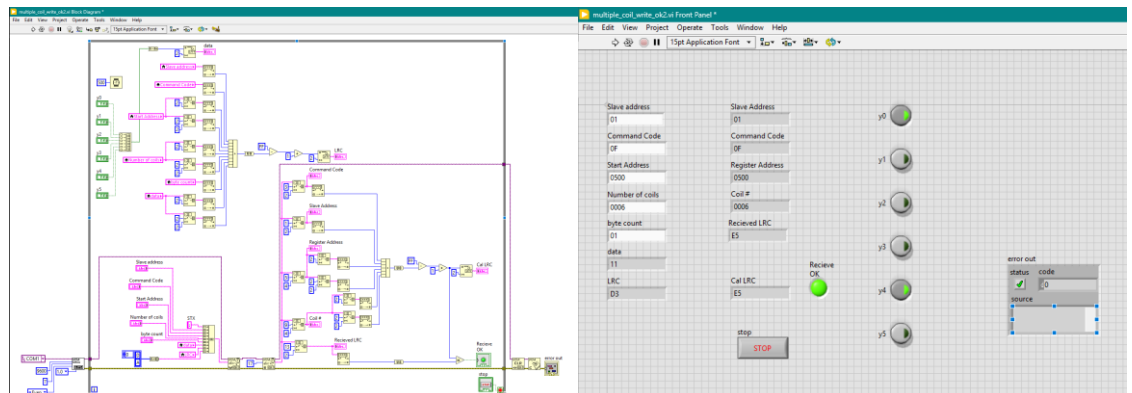
در جلسه ی قبلی برنامه ی Labview برای ارتباط مدباس Ascii با PLC دلتا DVP14SS2 نوشته شد، در این جلسه دستورات مختلف رو تست میکنیم.

- بررسی ساختار دستور 02 برای خواندن ورودی های دیجیتال PLC
- تغییر برنامه ی Labview برای ارسال دستور 02
- بررسی نحوه ی شماره گذاری کویل ها در PLC
- بررسی ساختار پاسخ دریافتی به دستور 02
- خواندن ورودی های دیجیتال PLC در Labview
- بررسی ساختار دستور 06 برای نوشتن مقدار رجیستر ۱۶ بیتی
- رجیسترهای 32 و ۱۶ بیتی و تفاوت دستورات LD و DLD در WPLSoft
- تغییر مقدار رجیستر ۱۶ بیتی در PLC با استفاده از Labview
- ساختار دستور 16 برای تغییر چند رجیستر ۱۶ بیتی
- اصلاح برنامه ی Labview برای ارسال دستور 16
- تست تغییر مقدار رجیسترهای ۳۲ بیتی PLC با Labview
- ساختار دستور 03 برای خواندن مقدار رجیسترهای PLC و اصلاح برنامه Labview
- بررسی پاسخ PLC به دستور 03 و اصلاح برنامه ی Labview
- تست پروژه خواندن رجیستر PLC با استفاده از Labview

کتابخانه آماده مدباس Ascii و RTU در Labview

در دو جلسه ی قبل بدون استفاده از کتابخانه ی آماده تمامی کدهای مورد نیاز برای مدباس Ascii در Labview رو نوشتیم. در این جلسه کتابخانه ی آماده ای که وجود داره رو بررسی میکنیم.

- آموزش نصب کتابخانه ی مدباس با VI Package manager
- بلوک create Modbus instance
- بلوک write multiple coil و تغییر خروجی های PLC
- تابع shutdown
- بلوک Read discrete input برای خواندن ورودی های دیجیتال PLC
- بلوک Read holding register برای خواندن رجیسترهای PLC
- تابع write multiple register
- تست برنامه مدباس RTU برای خواندن و نوشتن در بیت ها و رجیسترهای PLC



ارتباط مدباس Ascii و RTU بین PLC دلتا و میکروکنترلر STM32F103 به روش polling

در این فصل کتابخوه ارتباط مدباس Ascii و RTU در میکروکنترلر STM32F103 برای ارتباط با PLC دلتا DVP14SS2 از صفر مینویسیم.

ارتباط مدباس Ascii بین PLC دلتا و میکروکنترلر STM32F103 به روش polling

- ساخت استراکچر برای ذخیره سازی محتوای دستورات 02، 03، 15 و 16
- نوشتن تابع محاسبه ی LRC
- تابع sprint برای جایگذاری اعداد در رشته
- تعریف استراکچر برای دستورات مدباس RTU
- نوشتن تابع محاسبه CRC16 برای مدباس RTU
- نوشتن تابع سریعتر جایگزین تابع sprint
- تنظیمات UART و پین های مورد نیاز برای مبدل RS485 به TTL
- تعریف استراکچر برای ذخیره سازی پاسخ دستورات دریافت شده از PLC
- تابع sprintf برای تبدیل رشته به مجموعه ی اعداد
- نوشتن تابع سریعتر از sprint برای استخراج اعداد از رشته ی دریافتی
- ارسال دستورات خوندن و نوشتن بیت و رجیستر از میکروکنترلر به PLC با روش polling

ارتباط مدباس Ascii بین PLC دلتا و میکروکنترلر STM32F103 با DMA

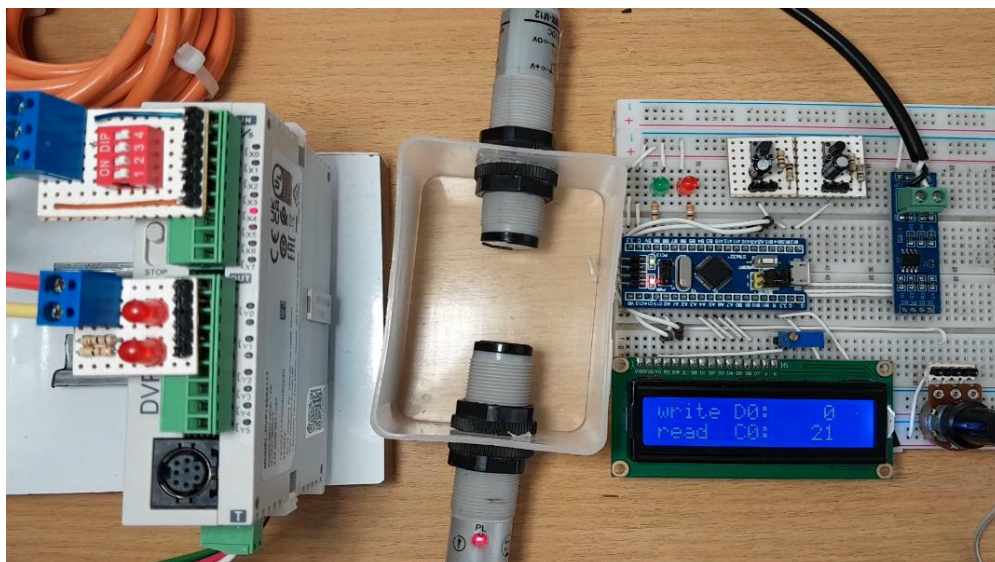
- بررسی زمان مورد نیاز UART برای ارسال دیتا
- بررسی تفاوت داده های دریافتی با DMA و بدون DMA و نوشتن تابع حذف parity
- استفاده از اینتراپت DMA برای آغاز عملیات ارسال بعدی مدباس
- تشخیص مرحله ی عملیات در هر اینتراپت با enum
- ارسال و دریافت دستورات مدباس Ascii از STM32F103 به PLC با استفاده از UART و DMA

ارتباط مدباس RTU بین PLC دلتا و میکروکنترلر STM32F103 با DMA

- اعمال تنظیمات ارتباط RTU در PLC با نرم افزار WPLSoft
- بررسی و تست زمانبندی مدباس RTU و اشتباهات رفرنس منوال PLC
- تغییر توابع و تعریف ها برای مدباس RTU
- اجرای مدباس RTU برای خواندن و نوشتن در بیت ها و رجیسترهای PLC توسط STM32F103
- ایجاد فاصله ی زمانی بین دریافت و ارسال بعدی با استفاده از اینتراپت تایمر در حالت تک پالس

ارتباط مدباس RTU بین HMI دلتا DOP107 و میکروکنترلر STM32F103 با روش polling و DMA

- کتابخانه مدباس رو برای HMI دلتا هم مینویسیم



دوره هنوز تکمیل نشده و همه ی پریفرال های میکروکنترلر STM32F103 به دوره ی پیشرفته اضافه میشه

I2c , USB , SPI , CAN