آزمایش ۱۰

آزمایشگاه ریزپردازنده

نیم‌سال اول ۱۴۰۲-۱۴۰۱

محممهدی چیذری ۹۷۲۴۳۰۱۸

سیدعباس میرقاسمی ۹۷۲۴۳۰۶۸

سوالات تحلیلی

۱. با مطالعه پروتکل سریال USART و بلوک مربوطه در STM32F401، پارامترهای قابل تنظیم آن را ذکر کنید.

این بلوک از میکرو را در مد های مختلف کاری راه اندازی کرد که اکثر پارامترهای هر مد مشترک هستند که به شرح زیر می باشد:

Baud Rate نرخ ارسال و دیافت دیتا

Word length تعداد بیت دیتای قابل استفاده در هر ارسال.

Parity تولید بیت پریتی یا خیر و زوج و فرد بودن آن

Sop Bit تعداد استاپ بیت برای هر داده

Data order ترتیب msb or lsb بودن

همچنین پارامتر های مختص هر مد نیز به این شرح می باشد: بیت های کنترلی CTS یا RTS یا هردو با هم برای اتصال غیرهمزمان دوسیستم. در مد غیر همزمان و اتصال تک سیمه، جهت حرکت داده و تعداد Oversampling از هر داده و در مد Multi Processor علاوه بر این موارد، پارامتر Wakeup method نیز قابل تنظیم است. در مد همزمان میتوان پارامتر های Clok polarity ، Clock phaseو Clock Last bit را و جهت داده رو میشه تنظیم کرد. در مد IrDA علاوه بر جهت داده، Power modeو prescaler و در مد LIN جهت داده و Break detect length رو میشه تنظیم کرد. در مد SmartCard پارامترهای جهت داده، Nack دادن در صورت parity error و guardtime قابل تنظیم هستند.

۲. نحوه فعال کردن وقفه دریافت داده‌های سریال UART را بیان کرده و رجیسترهای مربوطه را به طور کامل شرح دهید.

۱) مقداردهی به رجیسترهای CR و CFGR (یعنی منبع کلاک سیستم را HSI میزاریم)

۲) نوشتن ۱ بر روی UE در رجیستر USART_CR1 (یعنی واحد USART را فعال میکنیم).

۳) مشخص کردن تعداد بیتهای داده در USART_CR1

۴) در USART_CR1 ، REرا یک میکنیم تا بخش رسیور فعال شود. همچنین RXNEIE را نیز مقدار میدهیم تا در هنگام دریافت داده، درخواست وقفه ایجاد شود.

۵) مشخص کردن تعداد استاپ بیت در در USART_CR2

۶) با استفاده از فرمول زیر، مقادیر مرتبط با باودریت مد نظر را محاسبه میکنیم:
Tx / Rx baud = fck / (16 * USARTDIV)
و در رجیستر USART_BRR قرار میدهیم.
۷) بعد از اتمام دریافت دیتا توسط USART ، بیت RXNE در رجیستر USART_SR ست شده و وقفه رخ میدهد.
۸) در روتین وقفه، رجیستر USART_DR خوانده میشود تا دیتای دریافت شده بدست آید.

۳. کنترل خطا در ارتباط سریال USART و SPI با چه منطقی انجام می‌گیرد؟ شرح دهید.

کنترل خطا در هر دو spi , usart بوسیله status register انجام می شود. که در هر کدام بیت های با مفاهیم متفاوتی برای کنترل خطا وجود دارد که در ذیل توضیح خواهیم داد:

در usart:

Overrun error زمانی که داده ای جدید دریافت شود و هنوز داده قبلی خوانده نشده است، این بیت توسط سخت افزار تنظیم میشود.

Noise detected flag اگر نویز روی یک فریم دیافتی باشد سخت افزار این رو ۱ میکنه

Framing error در صورت نویز بیش از حد یا یک کاراکتر بریک این بیت ۱ میشه توسط سخت افزار

Parity error درصورت وجود خطای پریتی توسط سخت افزار یک میشه

در spi:

underrun flag اگر اولین ساعت برای انتقال داده ظاهر شود در حالی که نرم افزار هنوز هیچ مقداری را در SPI_DR بارگذاری نکرده است، این بیت یک میشه

Overrun flag: اگر داده ای دریافت شود در حالی که داده قبلی هنوز از SPI_DR خوانده نشده است یک میشود

Frame error flag اگر i2sدر حالتSlave پیکربندی شده باشدو masterخارجی، خط WS را در لحظه ای تغییر دهد که Slave انتظار این تغییر را نداشته باشد این بیت ۱ میشود.

و برنامه نویس باید هنگام برداشتن داده، صحت دریافت و ارسال داده رو بررسی کند.

۴. وظیفه کنترل داده‌ها و استخراج داده از بیت‌های کنترلی بر عهده برنامه‌نویس است یا ماژول سخت افزاری؟ شرح دهید که چرا از ماژول سخت‌افزاری برای این کار استفاده می‌شود؟

سخت افزاری. تمامی این کارها توسط سخت افزار و به صورت موازی با کارکرد پردازنده انجام میشود تا پردازنده فرصت انجام بقیه کارهای پردازشی را داشته باشد. حال برنامه نویس تنها لازم است که زمانی که سخت افزار داده را به طور کامل دریافت کرد، توسط وقفه، DMAیا به طور مستقیم در کد و با چک کردن بیت کنترلی، دیتای لازم را استفاده کند و توان پردازشی کنترلر را برای انجام عملیاتی بر اساس این داده استفاده کند، (نه برای دریافت آن). زیرا اینکار بصورت نرم افزاری عملا ممکن نیست، حتی زمانی که تمام عملکرد CPU و به صورت بهینه شده، صرف انجام این کار شود. سیستم باید بیت های مرتبط را آزمایش کند و به ازی هر flag، عمل مرتبط را انجام دهد. حتی اگر این عملیات را به بهینه ترین حالت ممکن و به زبان اسمبلی بنوییسم، میبینم که پردازنده زمانی برای انجام تمامی کارهای لازم درتعداد سیکل معقول ندارد. یا مثلا اگر دریافت یا ارسال یک داده USART را در نظر بگیریم، پردازنده باید دائما منتظر زمان مناسب باشد تا دیتای مناسب را یا بر روی line

قرار دهد یا از آن بخواند. حتی اگر اینکار توسط وقفه هم انجام شود، به خصوص در buad rate های بالا، پردازنده دائما در حال رفت و برگشت به وقفه های پی در پی است که صرف نظر از کد وقفه، رفتن و برگشتن از خود وقفه چندین سیکل ساعت طول میکشد. اگر قرار باشد نرم افزار بعد از اینکه یک فریم داده را دریافت کرد، بررسی کند که آیا دریافت به درستی صورت گرفته یا خطایی رخ داده است، عملا هیچ قدرت پردازشی اضافه ای برای بقیه انجام کارها باقی نمی ماند.