SPI

-Master chip select pins are called (NPCS), slave chip select pins are called (NSS)

-We can transmit 8 or 16-bit data per frame

We've to enable the SPI clock by configuring the PMC

Master mode main configurations:

- In the register SPI_MR ➔ set the MSTR bit to 1
- Configure the NPCS0, NPCS1, NPCS2, NPCS3 to be output pins
- Drive the SPI clock (SPCK)
    - configure the (CPOL) and (NCPHA) pins for the clock polarity and phase

Slave mode main configurations:

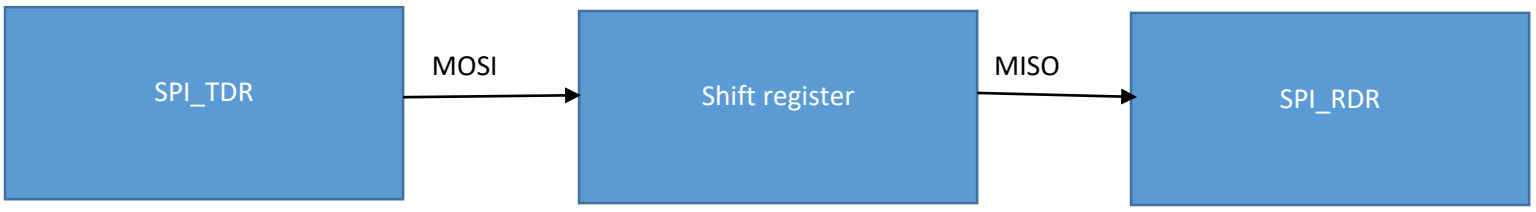
- SPI_MR ➔ clear the MSTR bit (make it 0)
- NPCS pins are configured as inputs

ال SPI فيه register مسئول عن ال data transfer و register مسئول عن ال data receive .. وفيه بينهم shift register بيبقى زي محطة بين ال 2 registers دول

فأول لما نعمل enable لل SPI وال processor يحط داتا في ال transfer register (elly hwa esmo SPI_TDR) ساعتها ال data transfer هيبدأ

دايماً في أي communication protocol الماستر هو اللي بي initiate ال communication

ال flow اللي بيحصل ان انا مثلاً ماستر وانت slave :P وانا عايز ابعتلك داتا، ساعتها انا (اللي هو البروسيسور) هحط الداتا اللي عايز ابعتهالك في ال register اللي اسمه TDR .. وانت في الناحية التانية معاك داتا في الريجستر اللي اسمه RDR .. فانا عشان انقل لك الداتا بتاعتي لازم انت كمان تنقلي الداتا بتاعتك (اللي هي غالبا بتبقى garbage ومالهاش لازمة) ب circular buffer.. فال spi هياخد الداتا بتاعتي اللي في ال TDR بتاعي ويحطها في ال shift register ويبدأ ينقل من ال shift register لل RDR بتاعك، وكل clock cycle فيه 1 bit بتتنقل، وفي نفس الوقت فيه 1 bit بتطلع من ال RDR عشان تفضي مكان لل bit اللي جاية جديد (غالبا بتروح لل TDR بس مش متأكد) .. وهكذا بقى لحد لما الداتا اللي عايز انقلهالك تخلص



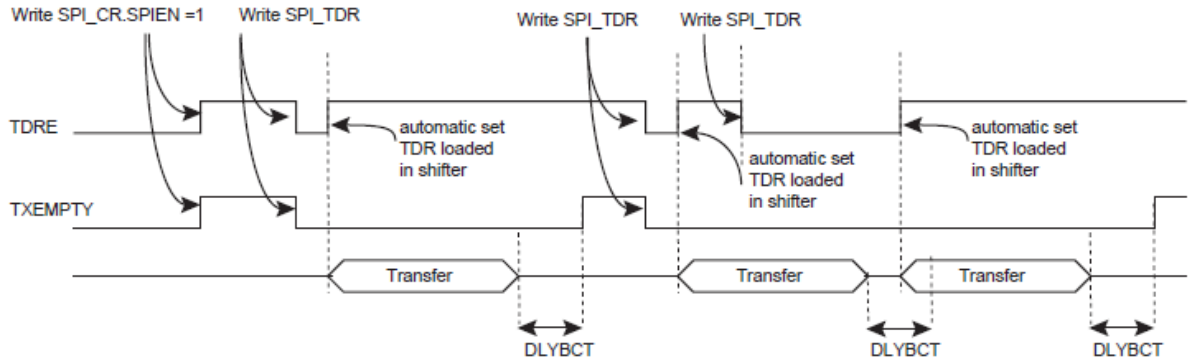
لو مافيش داتا في ال TDR وانا عملت set ل bit اسمها WDRBT في register اسمه SPI_MR .. ساعتها ال TDR هيتحط فيه وحايد .. **والمفروض** ان الماستر هيستنى لحد لما ال processor بتاع ال slave يقرا الداتا اللي جت له على ال RDR عشان يبدأ يعمل transfer جديد (عشان مايعملش override على الداتا اللي موجودة في ال RDR ولسة ماتقريتش)

-قبل ما نبدأ نكتب في ال TDR لازم نعمل set ل bits معينة اسمها PCS موجودة في ال register اللي اسمه SPI_MR عشان نحدد مين ال slave اللي عايزين نديله الداتا دي

لو حصل بقى ان في نص ال data transmission بقى فيه data جديدة دخلت في ال TDR وعايزة تدخل على ال shift register .. ساعتها هتستنى لحد لما ال transmission الاولاني يخلص الأول وبعد كدة الداتا الجديدة تبدأ تروح على ال shift register ويبدأ ال new transfer

.. طيب افرض فيه داتا واقفة مستنية في ال TDR ويشاء السميع العليم ان يبقى فيه داتا جديدة كمان عايزة تدخل على ال TDR.. لو دخلت على الداتا اللي موجودة هيحصل مصيبة، فانا في الكود مش هينفع اخليها تدخل غير لما اتأكد ان ال TDR فاضي .. هتأكد من كدة ازاي؟؟ هروح اقرا flag اسمه TDR register empty (TDRE) موجود في SPI_SR .. ال flag دة لو ب 1 يبقى ال register فاضي واقدر اكتب عليه، ولو ب 0 يبقى لازم استناه لحد لما يبقى ب 1 عشان ابقى في السليم .. وهو بيبقى 1 اول لما الداتا تخرج من ال TDR وتروح على shift register .. طيب دة بالنسبة لخروج الداتا من ال TDR لحد لما تروح لل shift register ... عايزين بقى flag تاني يقول لنا ان الداتا خلاص خرجت من ال shift register وراحت لل RDR .. دة بقى اسمه TXEMPTY

Figure 33-5. TDRE and TXEMPTY flag behavior



هتلاقوا فيه delay في الرسمة (هنجيبه قدام)

وهتلاحظوا ان اول ما ال SPI بيشتغل بتبقى ال TRDE & TXEMPTY = 1 لحد لما نبدأ نكتب اول داتا في ال TDR

فيه flag كمان برضه اسمه RDRF دة بيشوف لو ال RDR دة full او لأ

.. لو فيه داتا داخلة على ال RDR قبل ما الداتا القديمة اللي كانت فيها تتقري ساعتها هيحصل overrun .. ففيه flag اسمه overrun error (OVRES) بيبين ان المشكلة دي حصلت

- نقدر نعمل clock division لل spi عن طريق bits اسمها SCBR موجودة في SPI_CSR
- فيه delay بنحدده بيبقى بين ال chip selects .. اللي هو الديلاي اللي هيحصل لما احول من cs1 ل cs2 .. ودة بيتحدد على أساس اكبر ديلاي موجود لل slave .. اسمه DLYBCS وموجود في SPI_MR
- فيه ديلاي تاني بيبقى قبل ما ال spi clock تشتغل اسمه DLYBS وبيتحدد لكل CS
- ديلاي كمان بيبقى بين كل transfer والتاني وبيتبرمج برضه لكل CS .. بيعبر عن الوقت اللي ال slave بياخده عشان ي process the received data

المفاجأة بقى اننا ممكن نحط decoder على ال 4 NPCS pins وبالتالي هيطلع 16 pin فاحنا نقدر نوصل 15 peripheral بال spi .. مش 4 بس :D

وعشان نعمل كدة فيه bits اسمها PCSDEC بنقولها اننا هنعمل كدة عشان ال processor يبقى عارف

بس لو احنا مش عاملين كدة بنخلي ال bit دي ب 0 .. وساعتها ال processor مش هينفع يشوف اكتر من cs معمولها activation في نفس الوقت .. ولو دة حصل هو بيختار واحدة بيعمل لها activation ويفكس للتانية .. ودايماً هو بيختار الرقم الأقل .. يعني لو NPCS0 و NPCS1 هو هيختار NPCS0

وكل NPCS له register لوحده .. احنا معانا 4 CS ب 4 registers ... تبذير اوفر والله

فيه حاجة اسمها mode fault detection موجودة ل NPCS0 بس .. ودي بتشوف لو حصل في وقت معين و بقى فيه كذا ماستر (وطبعاً دة ماينفعش) فساعتها بي detect المشكلة دي

- افرض بقى انا عايز ادخل كذا داتا ورا بعضها على نفس ال CS .. احنا قلنا ان ال TDRE لما يبقى ب 1 ساعتها ال TDR هي load الداتا الجديدة ويبدأ ينقلها .. وبعد كدة ال CS is deactivated .. بس احنا مش عايزينها تتقفل دلوقتي احنا لسة عايزين ننقل حاجات تانية فاحنا عايزين حاجة تحافظ لنا على ال CS وماتخليهوش يتغير بمزاجه .. الحاجة دي هي bit اسمها chip select active after transfer (CSAAT) .. لما اعملها ب 1 انا كدة بقول لل processor انا عايز ال CS دي تفضل متأكتفة حتى لما ال transfer يخلص عشان يستنى الداتا الجديدة ... بس في الاخر في اخر transfer خالص محتاجين ن clear ال bit دي عشان ال CS دي ماتفضلش متأكتفة مدى الحياة كدة
- ففيه bit تانية اسمها Last transfer (LASTXFER) دي لما اخليها ب 1 بتعرف ان دة اخر transfer خلاص

فيه flow chart في صفحة 695 ملخص الدنيا بالنسبة لل master mode

تقريبا دة الملخص بتاع الكلام اللي في الداتاشيت .. افتحوا بقى صفحة 704 وشوفوا ال registers