We are interfacing it with STM32L476VGT Cube 4.26.1.  Our firmware is running on STM32Cube\_FW\_L4\_V1.12.0 (equivalent to 1.13.0 qspi). We have been able to successfully read the device ID (0x01 0x60 0x19), erase block (cant verify). Below are the routines, and the capture from our Saleae analyzer. I am looking for any guidance to help understand what the cypress part is looking for.

st\_ReadQspiID();                   // ensure we see a 25FL256

uint8\_t     st\_ReadQspiID(void)

{

    QSPI\_CommandTypeDef  sCommand;

    uint32\_t address = 0;

    uint16\_t index;

    \_\_IO uint8\_t step = 0;

    uint8\_t \*   rcv;

    rcv= &aRxBuffer;

    sCommand.InstructionMode  = QSPI\_INSTRUCTION\_1\_LINE;

    sCommand.AddressSize    = QSPI\_ADDRESS\_32\_BITS;

    sCommand.AlternateByteMode = QSPI\_ALTERNATE\_BYTES\_NONE;

    sCommand.DdrMode      = QSPI\_DDR\_MODE\_DISABLE;

    sCommand.DdrHoldHalfCycle = QSPI\_DDR\_HHC\_ANALOG\_DELAY;

    sCommand.SIOOMode     = QSPI\_SIOO\_INST\_EVERY\_CMD;

    sCommand.Instruction = READ\_ID\_CMD;  // 9Fh

    sCommand.AddressMode = QSPI\_ADDRESS\_NONE;

    sCommand.DataMode  = QSPI\_DATA\_1\_LINE;

    sCommand.DummyCycles = 0;

    sCommand.NbData   = 3;

    StatusMatch = 0;

    RxCplt = 0;

   sCommand.DummyCycles        = 0;

    if (HAL\_QSPI\_Command\_IT(&hqspi, &sCommand) != HAL\_OK)

    {

        Error\_Handler();

    }

    if (HAL\_QSPI\_Receive\_IT(&hqspi, rcv) != HAL\_OK)

    {

        Error\_Handler();

    }

    while (RxCplt == 0)

    {

        vTaskDelay(1);

    }

        if ((aRxBuffer[0] ==  CYPRESS)&&(aRxBuffer[1] == FL\_FLASH) && (aRxBuffer[2] ==FLASH\_256))

    {

        st\_ConsolePrint("Detected s25FL256L QSPI FLASH\n\r");

    }

    return(0);

}



    st\_ReadQspiConfig();

uint8\_t     st\_ReadQspiConfig(void)

{

    QSPI\_CommandTypeDef sCommand;

    uint32\_t address = 0;

    uint16\_t index;

    \_\_IO uint8\_t step = 0;

    uint8\_t \*   rcv;

    rcv= &aRxBuffer;

    sCommand.InstructionMode  = QSPI\_INSTRUCTION\_1\_LINE;

    sCommand.AddressSize    = QSPI\_ADDRESS\_32\_BITS;

    sCommand.AlternateByteMode = QSPI\_ALTERNATE\_BYTES\_NONE;

    sCommand.DdrMode      = QSPI\_DDR\_MODE\_DISABLE;

    sCommand.DdrHoldHalfCycle = QSPI\_DDR\_HHC\_ANALOG\_DELAY;

    sCommand.SIOOMode     = QSPI\_SIOO\_INST\_EVERY\_CMD;

    sCommand.Instruction = READ\_CONTROL\_2;

    sCommand.AddressMode = QSPI\_ADDRESS\_NONE;

    sCommand.DataMode  = QSPI\_DATA\_1\_LINE;

    sCommand.DummyCycles = 0;

    sCommand.NbData   = 1;

    StatusMatch = 0;

    RxCplt = 0;

   sCommand.DummyCycles        = 0;

    if (HAL\_QSPI\_Command\_IT(&hqspi, &sCommand) != HAL\_OK)

    {

        Error\_Handler();

    }

    if (HAL\_QSPI\_Receive\_IT(&hqspi, rcv) != HAL\_OK)

    {

        Error\_Handler();

    }

    while (RxCplt == 0)

    {

        vTaskDelay(1);

    }

    return(0);

}



    st\_QFLASH\_WriteExtendAdd(&hqspi);  // put device in 32 bit addressing mode

/\*\*

  \* @brief  This function enables Extended Addressing on memory side.

  \* @param  hqspi: QSPI handle

  \* @retval None

  \*/

void st\_QFLASH\_WriteExtendAdd(QSPI\_HandleTypeDef \*hqspi)

{

  QSPI\_CommandTypeDef sCommand;

    /\* Enable write operations ---------------------------------------- \*/

  QSPI\_WriteEnable(hqspi);

  /\* Write BAR or Status Register  --------------------------- \*/

  sCommand.InstructionMode   = QSPI\_INSTRUCTION\_1\_LINE;

  sCommand.Instruction       = ENTER\_4BYTE\_ADD;

  sCommand.AddressMode       = QSPI\_ADDRESS\_NONE;

  sCommand.AlternateByteMode = QSPI\_ALTERNATE\_BYTES\_NONE;

  sCommand.DataMode          = QSPI\_DATA\_NONE;

  sCommand.DummyCycles       = 0;

  sCommand.DdrMode           = QSPI\_DDR\_MODE\_DISABLE;

  sCommand.DdrHoldHalfCycle  = QSPI\_DDR\_HHC\_ANALOG\_DELAY;

  sCommand.SIOOMode          = QSPI\_SIOO\_INST\_EVERY\_CMD;

  sCommand.NbData            = 1;

  if (HAL\_QSPI\_Command\_IT(hqspi, &sCommand) != HAL\_OK)

  {

    Error\_Handler();

  }

  wait\_for\_qspi(hqspi);

}



    st\_QFLASH\_WriteQPI(&hqspi);

/\*\*

  \* @brief  This function enables QPI Mode on memory side.

  \* @param  hqspi: QSPI handle

  \* @retval None

  \*/

void st\_QFLASH\_WriteQPI(QSPI\_HandleTypeDef \*hqspi)

{

  QSPI\_CommandTypeDef sCommand;

  uint8\_t reg= EXTADD\_BIT;

    /\* Enable write operations ---------------------------------------- \*/

  QSPI\_WriteEnable(hqspi);

  /\* Write BAR or Status Register  --------------------------- \*/

  sCommand.InstructionMode   = QSPI\_INSTRUCTION\_1\_LINE;

  sCommand.Instruction       = ENTER\_QPI;

  sCommand.AddressMode       = QSPI\_ADDRESS\_NONE;

  sCommand.AlternateByteMode = QSPI\_ALTERNATE\_BYTES\_NONE;

  sCommand.DataMode          = QSPI\_DATA\_NONE;

  sCommand.DummyCycles       = 0;

  sCommand.DdrMode           = QSPI\_DDR\_MODE\_DISABLE;

  sCommand.DdrHoldHalfCycle  = QSPI\_DDR\_HHC\_ANALOG\_DELAY;

  sCommand.SIOOMode          = QSPI\_SIOO\_INST\_EVERY\_CMD;

  sCommand.NbData            = 1;

  if (HAL\_QSPI\_Command\_IT(hqspi, &sCommand) != HAL\_OK)

  {

    Error\_Handler();

  }

  if (HAL\_QSPI\_Transmit\_IT(hqspi, &reg) != HAL\_OK)

  {

    Error\_Handler();

  }

  wait\_for\_qspi(hqspi);

}



   st\_EraseBlock25FL256L(0);

I had to include two analyzer shots to show but looks like it completed

///////////////////////////////////////////////////////////

uint8\_t st\_EraseBlock25FL256L(uint16\_t block\_no)

{

    QSPI\_CommandTypeDef sCommand;

    uint32\_t address = block\_no \* BLOCK\_SIZE;

    uint16\_t index;

    sCommand.InstructionMode   = QSPI\_INSTRUCTION\_1\_LINE;

    sCommand.AddressSize       = QSPI\_ADDRESS\_24\_BITS;

    sCommand.AlternateByteMode = QSPI\_ALTERNATE\_BYTES\_NONE;

    sCommand.DdrMode           = QSPI\_DDR\_MODE\_DISABLE;

    sCommand.DdrHoldHalfCycle  = QSPI\_DDR\_HHC\_ANALOG\_DELAY;

    sCommand.SIOOMode          = QSPI\_SIOO\_INST\_EVERY\_CMD;

        CmdCplt = 0;

        /\* Initialize Reception buffer --------------------------------------- \*/

          memset(aRxBuffer,256,0);

        /\* Enable write operations ------------------------------------------- \*/

        QSPI\_WriteEnable(&hqspi);

        /\* Erasing Sequence -------------------------------------------------- \*/

        sCommand.Instruction = SECTOR\_ERASE\_CMD;

        sCommand.AddressMode = QSPI\_ADDRESS\_1\_LINE;

        sCommand.Address     = address;

        sCommand.DataMode    = QSPI\_DATA\_NONE;

        sCommand.DummyCycles = 0;

        if (HAL\_QSPI\_Command\_IT(&hqspi, &sCommand) != HAL\_OK)

        {

          Error\_Handler();

        }

        if(CmdCplt != 0)

       {

          CmdCplt = 0;

          StatusMatch = 0;

          /\* Configure automatic polling mode to wait for end of erase ------- \*/

          QSPI\_AutoPollingMemReady(&hqspi);

        }

        if(StatusMatch != 0)

        {

          StatusMatch = 0;

          return(1);

        }

        else

        {

         return(0);

        }

}





    st\_QFLASH\_ReadPage(0 , &st\_LogCache);

void st\_QFLASH\_ReadPage(uint32\_t page ,uint8\_t \* data\_rec)

{

    QSPI\_CommandTypeDef sCommand;

    uint32\_t address = 0;

    uint16\_t index;

    \_\_IO uint8\_t step = 0;

    sCommand.InstructionMode   = QSPI\_INSTRUCTION\_1\_LINE;

    sCommand.AddressSize       = QSPI\_ADDRESS\_32\_BITS;

    sCommand.AlternateByteMode = QSPI\_ALTERNATE\_BYTES\_NONE;

    sCommand.DdrMode           = QSPI\_DDR\_MODE\_DISABLE;

   sCommand.DdrHoldHalfCycle  = QSPI\_DDR\_HHC\_ANALOG\_DELAY;

    sCommand.SIOOMode          = QSPI\_SIOO\_INST\_EVERY\_CMD;

    sCommand.DummyCycles       = DUMMY\_CLOCK\_CYCLES\_READ\_QUAD;

    CmdCplt = 0;

    StatusMatch = 0;

    RxCplt = 0;

   /\* Configure Volatile Configuration register (with new dummy cycles) \*/

   QSPI\_DummyCyclesCfg(&hqspi);

  /\* Reading Sequence ------------------------------------------------ \*/

          sCommand.Instruction = QUAD\_INOUT\_FAST\_READ\_CMD;

          sCommand.DummyCycles = DUMMY\_CLOCK\_CYCLES\_READ\_QUAD;

          sCommand.AddressMode = QSPI\_ADDRESS\_4\_LINES;

//         sCommand.Address= (page \* PAGE\_SIZE);

          sCommand.Address= 0x00;

          sCommand.DataMode          =   QSPI\_DATA\_4\_LINES ;

          sCommand.NbData=PAGE\_SIZE;

          if (HAL\_QSPI\_Command\_IT(&hqspi, &sCommand) != HAL\_OK)

          {

            Error\_Handler();

          }

    while (CmdCplt == 0)

    {

        vTaskDelay(1);

    }

         if (HAL\_QSPI\_Receive\_IT(&hqspi, data\_rec) != HAL\_OK)

          {

            Error\_Handler();

          }

       while (RxCplt == 0)

    {

        vTaskDelay(1);

    }

}



Further down the data changes from 0x8080 to 0x0000

