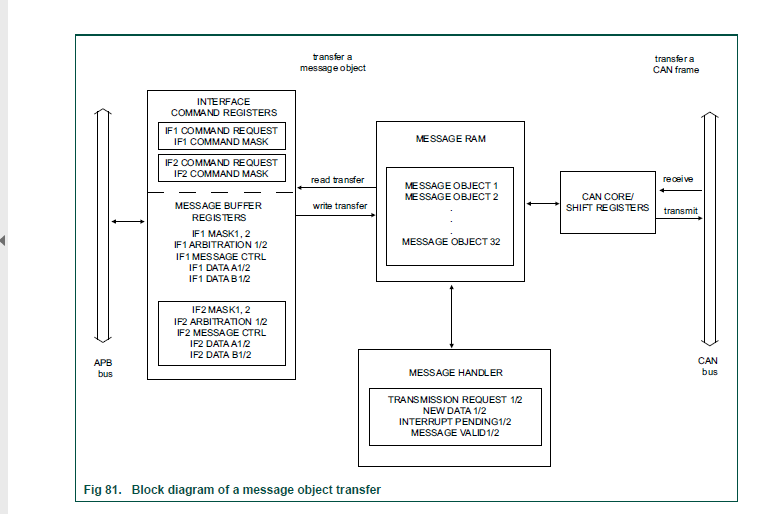
LPC1549的CAN 控制器的配置

Controller Area NetWork 结构框图



这里有 Message Interface 寄存器和 APB 总线相接，负责和CPU进行运行时数据交互，

Message RAM 负责和 Message interface 交互，同时负责和CAN\_CORE/SHIFT REGISTERS 交互。而CAN\_CORE/SHIFT REGISTERS 负责和CAN bus 交互。

# 一：初始化CAN控制器

每次reset 后，**CAN控制器**的**大部分寄存器**都保持**不变**，CAN控制器的 CAN control register 的 **INIT** 被置位为 1，**使能CAN控制器**的**软件初始化**。并且 busoff 位被置零， CAN\_TXD为高电平。由于使能了软件初始化，这个时候CPU可以对CAN 控制器进行初始化。

每当 INIT 被置位时，就可以对CAN控制器进行软件初始化，初始化期间**暂停和CAN BUS的通信**，**CAN\_TXD** 引脚为**高电平**，EC（error counter）**不计数**，配置寄存器保持不变，如果**CCE位被置高**，bit timing register 和 BRP extension register 就可以**被读写**修改。

当设置完毕后，通过把 INIT 位清零，BSP 同步后，等待 CAN\_TXD 的 11 位连续高电平后（CAN BUS 空闲），接入总线，开始进行数据传输。

# 二：配置 Message Objects

配置Message Object ，配置该片寄存器时，只需要把MSAGVAL 设置为 valid。在初始化时配置 Message Object ，需要把所有的Message Object 配置成特定格式，或者都设置成为not valid**。配置 Message Object** 是通过**配置两个** **interface register** 之一的MASK，ARBITRATIO， CONTROL，DATA filed来完成的。**配置完这些数据后**，通过**写 IFx Command Request Register** 把 **IFx Message Buffer Registers** 中数据**同步**到Message RAM中特定的 Message Object 里。

# 三：控制器运行

**数据接受：**

一旦初始化完成后，CAN 控制器会自己接上 can bus。接受到数据后，如果数据通过了 Message Handler’s 的过滤处理，数据会被存在合适的 Message Object 中。CPU 可以通过 Message Interface Registers 读取该些数据。

**数据发送：**

等待传送的数据由CPU准备好，如果数据的特定filed存在一个**配置好的特定**的（arbitration，控制位在**初始化期间**配置过）Message Object，那么只有数据 filed 会被传输过去，然后 **TXRQUT 和 NEWDAT** 位都会被置位，然后**开始传输**过程。

# 四：TXRQST 和 NEWDAT

一旦开始发送，**TXRQST 就会被清零**，然而 NEWDAT 保持置位直到数据发送完毕后，清零。所以**当一次发送失败**（失去 **arbitration或者发生错误**）时，**NEWDAT**仍然保持为高，这时候**为了重启发送**，需要**再次置位 TXRQST**。

# 五：CAN message handler

CAN message handler 负责处理CAN\_CORE 和 RX/TX Shift Register之间的数据传输过程，还有 Message RAM 和 IFX Registers 的数据传输过程。有如下四种功能。

1：从 Message RAM传输数据到 shift registers

2：从 shift registers传输数据到 Message RAM。

3：在 IFx Registers 和 Message RAM 之间传输数据。

4：从 shift registers 传输数据到数据过滤单元。

5：扫描 Message RAM 中对象以匹配特定 Message Object。

6：处理 TXRQST 标志。

7：处理中断。

一旦 INIT 位被置位，那么**CAN 控制器的状态机**和 **Message Handler** **状态机**控制着整个 CAN controller 内部的**数据流**。接受**通过过滤器的数据**，发送数据。CPU 通过 IFx Interface Registers 读取接受到的数据和发送数据。

# 六：在IFx Registers 和 Message RAM

当cpu 准备好在 IFx Registers 和 Message RAM 直接传数据时， Message Handler 会把对应的CMD 寄存器的 BUSY 位设置为 1。当传输完毕后再重置为 0。由于 Message RAM 的结构原因，不能够对 Message Object 进行单独的位或者字节传输，每次传输只能够写整个 Message Object，所以为了只修改某些位，而不影响其他位，只能够进行 read – modify – write 这样的步奏，才能完成数据传输。

先利用 CMDMASK 寄存器读出不需修改的 Message Object，这里没有被 CMDMASK 选中的将会保持不变。

把 Message buffer 中数据写入 Message Object，此时没有被 CMDMASK 的部分会保持Message Object 中本来的数据。