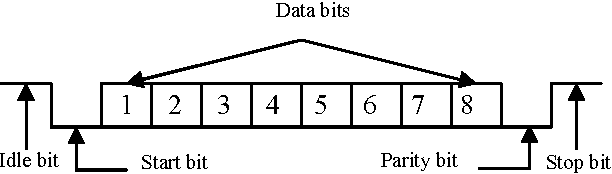


Content:

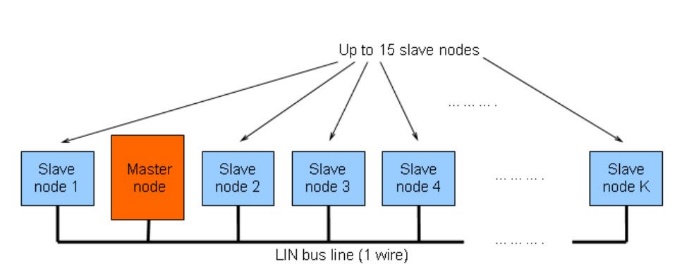
1. lin introduction
   1. local inter-connect network
   2. low cost low speed
   3. UART based
   4. Single wire
   5. Single master multi slave
   6. Broad cast
   7. 16 node
   8. Self-synchronization
   9. Message ID
   10. Collision detection
2. lin operation concepts
   1. master task (header)
   2. slave task (response)
3. lin frame structure
   1. header
      1. sync break
      2. sync byte
      3. message ID
         1. data length
         2. message priority
         3. parity bits
   2. response
      1. data
      2. check sum
   3. inter byte space
   4. response space
4. lin communication type
   1. data from master to slave
   2. data from slave to master
   3. data from slave to slave
5. lin frame type
   1. unconditional frame
   2. event triggered frame
   3. sporadic frame
   4. diagnostic frames
6. lin bus timing and scheduling table
   1. lin is time triggered protocol
   2. calculate frame time
7. lin error handling
   1. parity bits
   2. check sum
   3. application solution
8. lin sleep and wake up
   1. sleep
   2. wake up
9. lin connections in automotive ECU

|  |
| --- |
| 1. **LIN introduction** |

* اختصار local inter-connect network
* عملاه VOLVO و هدفه انه يبقى low cost , low speed عشان اوصله مع ال sensors و ال actuators و ال sensors اصلا بتبعت data قليله فمش محتاج ابعت data ب rate كبير
* ال license بتاعته free
* ال Lin هوا UART based protocol ( يعنى لما يبعت اى byte بيبعتها بشكل ال UART يعنى بيبعت start bit و بعديها ال byte و stop bit )



* ال lin هوا single wire bus
* Single master multi slave



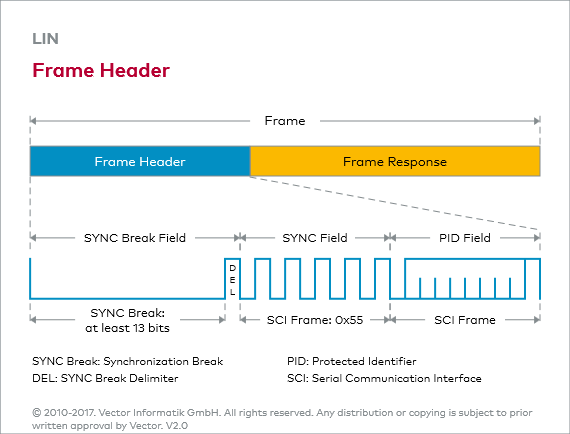
* ليه single master ؟ ...

عشان هوا بي communicate with sensors , actuators و دول مش منطقى انهم يبقو master دايما ال ECU هيا الى master

* شغال broad cast يعنى ممكن يبعت message و كل ال slaves يستقبلوها
* عدد ال nodes (node is a device connected on bus) ... 16 node
* Max speed 20 K bit/sec
* ليه مكان ال UART ؟

عشان مشاكل ال UART ذى

* + ال UART كان بيتكلم مع device واحد
  + مفيش error checking فى ال UART غير ال parity bit (ال lin بيستخدم ال check sum error checking )
  + اخره يبعت 1 byte ... (انما ال lin يقدر يبعت 1 or 2 or 8 byte)
* ال lin عنده self-synchronization يعنى ايه ؟

يعنى بيظبط ال clock الى هيتكلم بيها مع باقى الناس ... ازاى ؟

بيبعت byte فى اول كل frame عباره عن 0X55 ... اشمعنا ؟ عشان كلها 01010101 فالى يستقبلها يقدر منها يفهم ال master هيتكلم بانهى speed ال byte دى اسمها synchronization field بالتالى انا مش محتاج ابعت clock بالتالى هوا single wire

* ال LIN بيعتمد على ال message ID ... يعنى ال master بينزل على ال bus ... message ب address معين ... الى مهتم من ال slaves يقرا ال message دى يقراها و الى مش مهتم ... عنه ما قراها

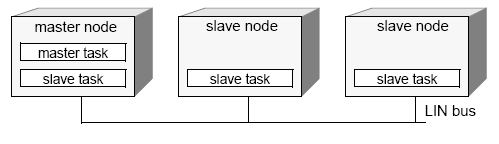
فانا بظبط ال configuration بتاعت ال slaves انهم يقرو ال message دى ( و دا بيقلل ال bandwidth لانى مش هبعت ال message اكتر من مره ذى ما كنت بعمل فى ال I2C , SPI )

* مفيش collision detection ... لان دايما ال master هوا الى هيبعت ... فعمر ما هتيجى case ان اتنين device يتكلمو فى نفس الوقت )

|  |
| --- |
| * افتكر ان ال * 0 -> dominant bit * 1 -> recessive bit * و دايما فى اى communication protocol ال idle = 1 عشان لما يجى حد يتكلم يبعت 0 فال 0 هوا الى يظهر على ال bus |

|  |
| --- |
| 1. **Operation concept** |

ال LIN frame لما بيجى يتبعت ... مش بيبعته حد واحد ... بيتشارك فى تكوينه 2 tasks ... يعنى ايه ؟



كل node فيها حاجه اسمها tasks ... يعنى مثلا

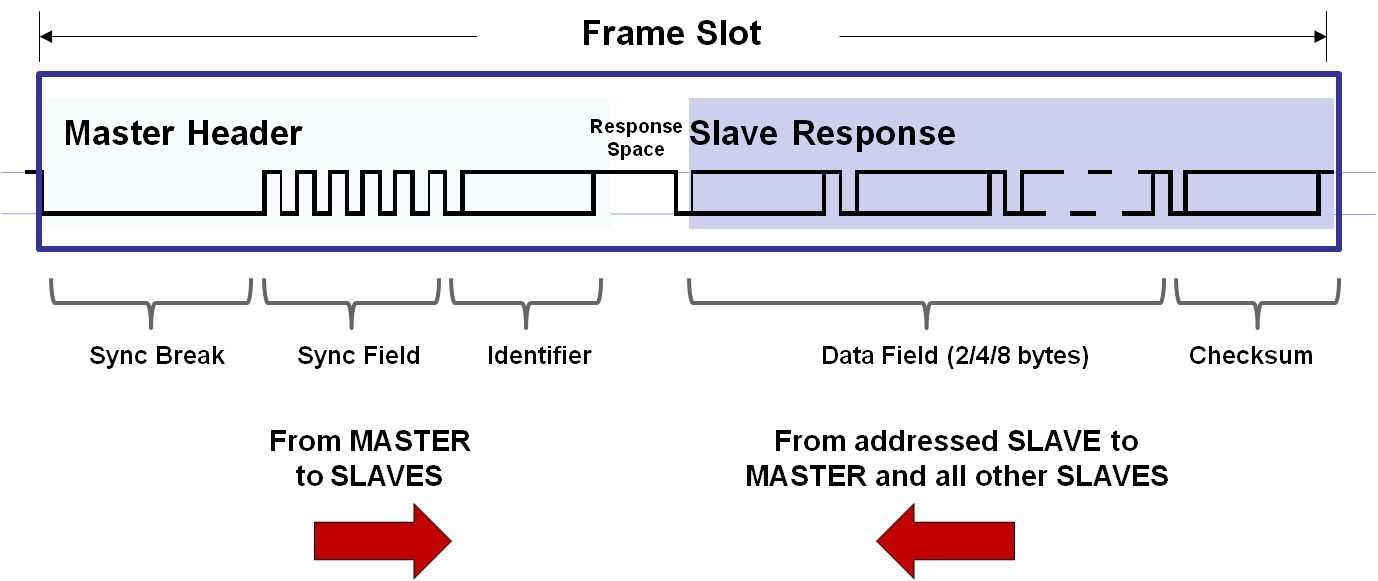
ال master node فيها 2 tasks

* Master task
* Slave task

و ال slave node عندها

* Slave task بس

و عشان ال frame يتكون ... بيتقسم لجزئين ... واحد بيكوننه ال master task و اسمه master header و التانى بيكوننه ال slave task سواء الى فى ال master او ال slaves و اسمه slave response



دايما اول جزء فى ال frame ال master هوا الى بيكوننه

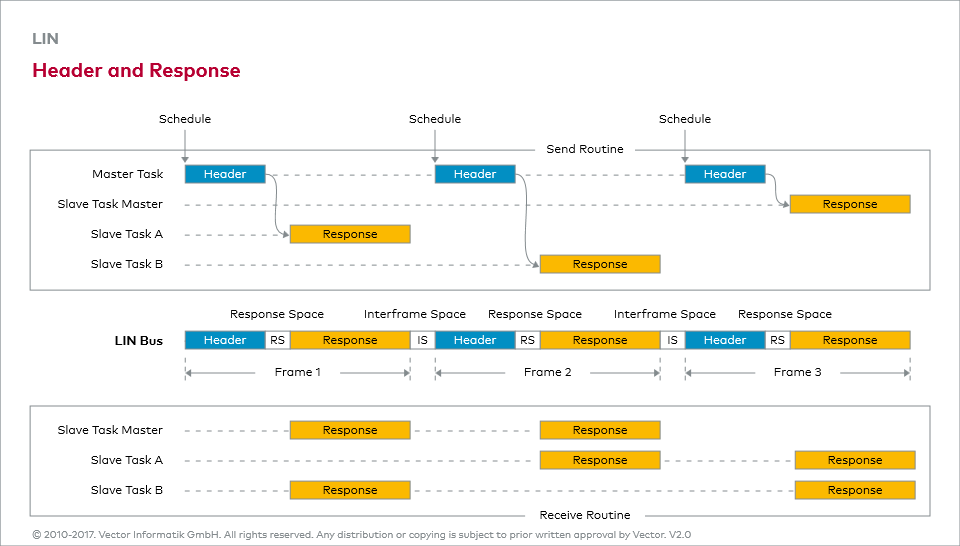
1. Master task

* دايما موجود فى ال master node
* دايما اول واحد بيشتغل ... لانه اول واحد هيبعت
* جواه 3 حاجات
  + Sync break
  + Synch field
  + Identifier (message ID)
* ال master task بيتحكم فى ...
  + ال bus عليه message ولا لا
  + بيتحكم فى ال communication لانه بيبعت اول نص ال frame
  + بيحدد ال message priority ... لانها بتعتمد على ال message ID
  + بيحدد سرعه ال transmission لانه بيبعت ال sync byte
  + بيستقبل ال wake up requests الى جايه من ال slaves

1. Slave task

* موجوده جوا ال master node و جوا كل slave node
* مسؤله عن تكوين النص التانى من ال frame الى اسمه response ال response بيتكون من ال
  + Data
  + Check sum

Message scenario :

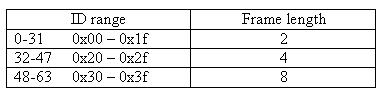


1. ال master بينزل على ال bus ال header ... الى جواه ال message ID
2. ال header لما ينزل يروح لكل slave task على ال bus سواء كانت slave task جوا ال slave node او حتى جوا ال master node ( هيا بتروح لكل slave task على ال bus )
3. على حسب ال message ID كل slave task معملوها configuration انها تعمل حاجه من 3
   1. Slave task -> send data
   2. Slave task -> receive data
   3. Slave task -> do nothing

* مثلا اول frame (من ناحيه الشمال ) الى كوننه ال master task فى ال master node و ال slave task الى فى node A (slave task A) كان معمولها configuration انها لما تشوف ال message ID دا ( الى اتبعت فى ال header ) ... ترد هيا بال response
* تانى frame كان ID مختلف و ال slave task الى فى node B هيا الى كانت المفروض تطلع ال response
* تالت frame الى طلع ال header هوا ال master task in master node و الى طلع ال response هوا ال slave task فى ال master node

|  |
| --- |
| 1. **lin frame structure** |

* 1. header
     1. sync break
* هدفه : ان ال master بيصحى الناس الى على ال bus عشان هيبعت header
* عباره عن 13 bit = 0 including the start bit
* اخرها break delimiter = 1bit = 1
* ليه 13 bit ؟ عشان ال slaves ياخدو وقتهم على ما يصحو
  + 1. sync byte
* هدفها : تظبط ال speed
* عباره عن 0X55 و اولها start , stop برضو
  + 1. message ID
* هدفه : عشان كل slave task تعرف ال action الى هتعمله based على ال ID دا سواء هتبعت او هتستقبل او مش هتعمل حاجه ... بيسموه protective ID
* هوا 8 bit ... بس ال ID فيهم 6 bit بس ... يعنى اخرى يبقى عندى 64 message
  + من 0-59 دول messages عاديه بتشيل data
  + 60 , 61 اسمهم diagnostic frames
  + 62 , 63 for future use
* ال message ID بيحددلى ...
  + Frame length



* + Priority
  + Parity bits

ال 2bits الى باقيين دول parity bits

* + اول parity bit عباره عن XOR(0,2,4)
  + تانى parity bitعباره عن XOR(1,3,5)
  1. Response
     1. Data
* بتتكون من 2 or 4 or 8 bytes على حسب ال message ID
* كل byte فيهم اولها start bit و اخرها stop bit ذى ال UART
  + 1. Check sum
* بيجمع ال data و وهوا بيبعت بيبعت ال summation
* ال receiver يستقبل ال data و يبعت ال check sum و يقارنه بال check sum الى جاله
* ال check sum الى ال LIN شغال بيه اسمه CRC(cyclic check sum)

الى بيتبعت ال inverted check sum

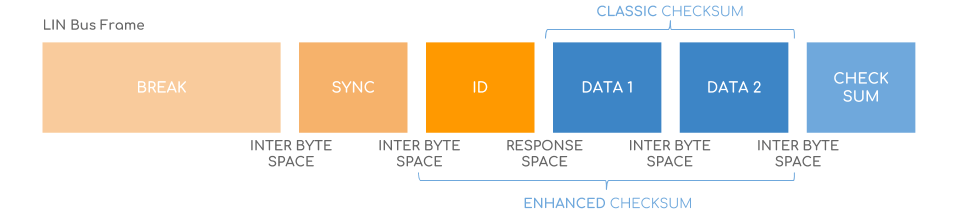
ال receiverبيحسب ال check sum و يجمعه مع ال inverted لو طلع 0xFF يبقى

كدا ال data جايه صح

* 1. Inter byte space

دى وقت بيتحط بين ال bytes عشان لو حصل شويه delay او عشان يلحق ال sensors تعمل process على ال byte لان برضو ال sensors هيا low cost فال accuracy بتاعتها مش كويسه

فممكن مثلا ال sensor يتاخر 3 or 4 bits مثلا **فانا ممكن اسمح بتاخير بين كل byte و التانيه**



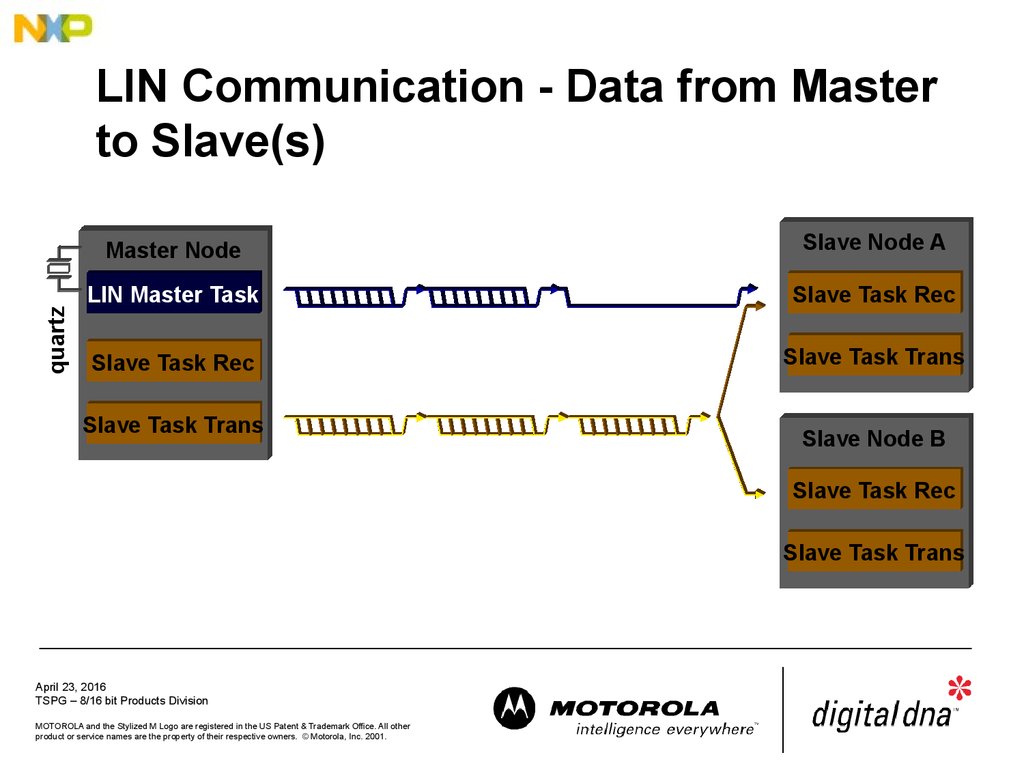
* 1. Response space
* دا delay بين ال header و ال response
* محتاجه ليه ؟ ... لان لما ال master ينزل header كل slave task المفروض تقرا ال message ID و تقرر على اساسه هل هتبعت response ولا هتستنى response من حد تانى ولا مش هتعمل حاجه

عشان تاخد القرار دا بتاخد وقت ... فلازم يبقى فى delay بين ال header و ال response على ما ال sensors slave tasks تقرر

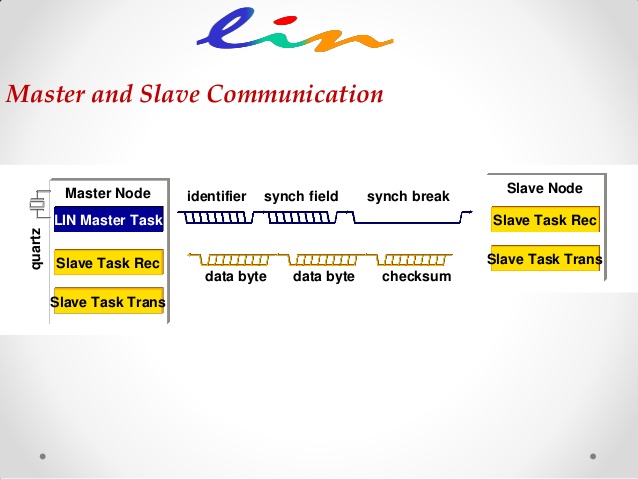
* الوقت دا بيقول قليل جدا لو ال slave task الى عند ال master node هيا الى هتبعت ال response

|  |
| --- |
| 1. **lin communication type** |

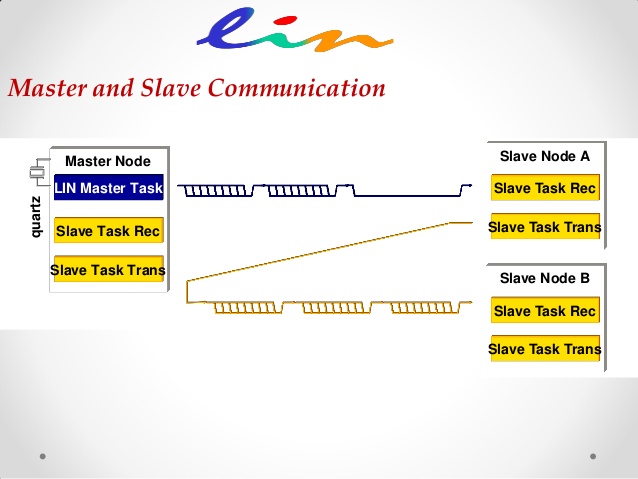
* 1. data from master to slave



* 1. data from slave to master



* 1. data from slave to slave



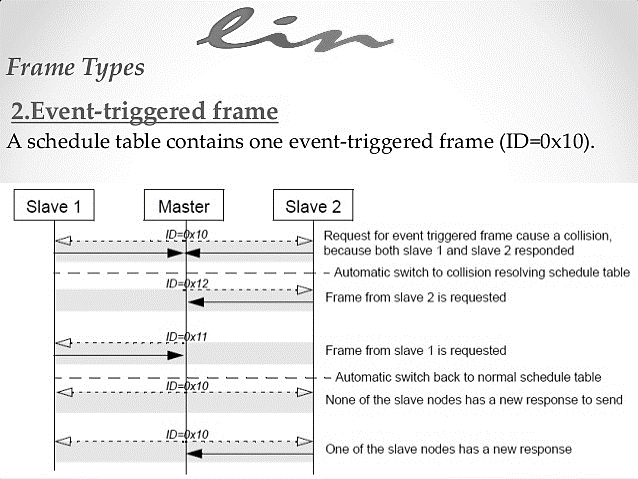
و فى كل الحالات ال master هوا الى بيبدا ال communication بانه يبعت ال header

|  |
| --- |
| 1. **lin frames types** |

* 1. unconditional frames (data frames)
* بيشيل message ID من 0-59
  1. Event trigger frame
* بيشيل برضو من 0-59
* هدفه : يقلل استخدام ال bus ... ازاى ؟

انى هخلى ال slave ميردش على ال master غير لما يكون فى عنده update فى ال data ... مثلا قرايه السينسور اتغيرت ... ساعتها يرد ... انما طول ما هيا ذى ما هيا هيرد ليه

بالتالى اوفر ال bandwidth



* مثلا ال master بعت header عشان يعرف حاله الابواب ... ف الباب الى عنده update بس هوا الى هيرد

طب لو الاتين ردو فى نفس اللحظه ... الmaster هيعتبره collision فيبتدى يكلم واحد واحد ب unconditional frame و كل واحد كلمه ب ID لوحده

كل ما كبر ال ID كل ما ال priority بتبقى اعلى

* 1. Sporadic frames

ذى ال event trigger بس لو ال master هوا الى هيبعت ال frame كله

يعنى ال master هيبعت ال header ... بس امتى هينزل ال response لو عنده update لو معندوش مش هيعمل حاجه ... فباقى ال slaves تفهم ان طالما ال master مبعتش response جديد يبقى هيا نفس ال data القديمه

و برضو هدفه يقلل ال band width

* 1. Diagnostic frames
* دى الى هيا ال commands ذى ال wake up command or sleep command

|  |
| --- |
| 1. **lin bus timing and scheduling table** |

LIN is time triggered protocol

* ال LIN هوا time triggered protocol يعنى كل frame ليها وقت معين تنزل فيه على ال bus

مش ذى باقى ال protocols وقت ما احب ابعت هبعت

* بالتالى هوا عنده schedule table فيه وقت كل header و بالتالى لازم ابقى عارف كل LIN frame وقته قد ايه عشان اعرف هبعت ال frame الى بعده امتى

وقت ال frame

Calculate frame time

Header =

sync break (13 bit) + delimiter (1 bit) +

sync byte with start , stop(10bit) + ID with start , stop (10bit)

= 34 bit

Response = data ((2 or 4 or 8 byte) + check sum (1byte) )\*10

Example : For 8-byte data

Bit number = 34 + (8 + 1)\*10 = 124 bit

و نضيف ال response space , interbyte space بالتالى نضرب 40% factor الى هيا 1.4

و فى العربيه ال header هينزل مثلا على ال bus كل 10 ms مثلا ... عشان اcheck الابواب ... على حسب نوع ال frame الابواب بقى هتشوف

* هترد على طول
* لو فى update هترد ... لو مفيش مش هتعبر ال master

|  |
| --- |
| 1. **lin error handling** |

* ال error handling مش جزء من ال LIN ... يعنى لو حصل error هيقوللى ان فى error بس عن طريق FLAG بس مش هيعمل action
* ايه ال errors الى ال LIN بي check عليها ؟
  + Parity bits (2 bit in header) check error in ID
  + Check sum (1byte in response) check for data , ID
* ال receiver هوا الى بيشوف ال هوا الى بيشوف ال errors دى ... بس ال master ميعرفش ان حصل error
* الحل ؟

انى اعمل message ب ID معين ... لو بعتته ال slave يرد عليه بقيمه ال flag error bits الى عنده و بناء عليه ال master يشوف هيعمل ايه

|  |
| --- |
| 1. **lin sleep and wake up** |

sleep

* ممكن ال LIN bus كله بكل ال nodes الى عليه ادخلهم فى sleep mode
* هيخش فى ال sleep mode امتى ؟
  + لو ال bus فضل idle اربع ثوانى
  + لو ال master بعت diagnostic frame (sleep command) with ID = 60

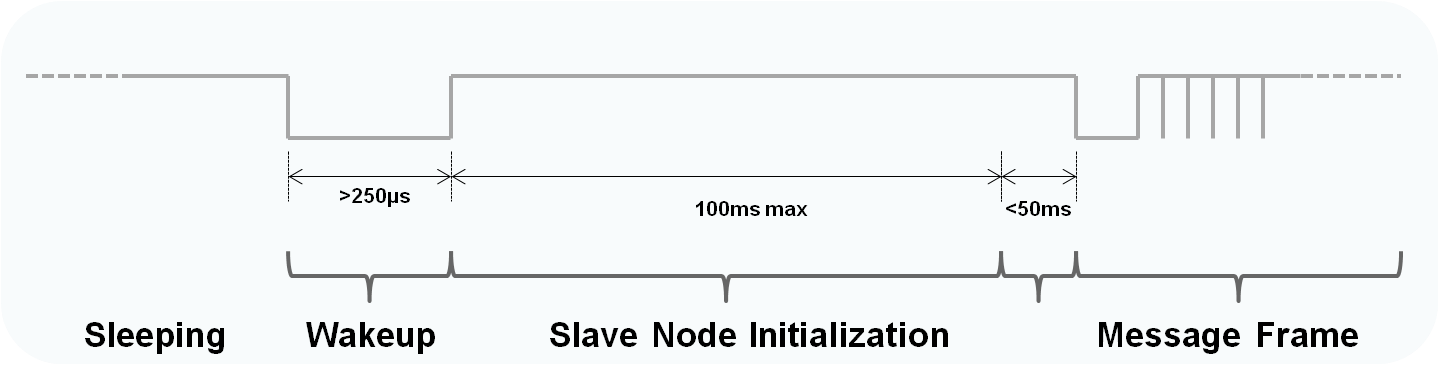
ال data فيها 8 byte اول byte = 0 و الباقى كله 0XFF

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data | | | | | | | |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| 0x00 | 0xFF | 0xFF | 0xFF | 0xFF | 0xFF | 0xFF | 0xFF |

* الهدف : عشان يقلل ال power consumption

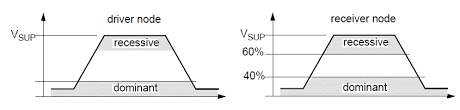
Wake up

* المسؤل عن ال wake up هوا ال slave task لانها بتبقى عايزه تبعت data فى وقت معين مثلا بس لازم ال تصحى ال master task عشان يبعتلها ال header
* ازاى ؟



1. ال slave يبعت 0 من 250 micro و يستنى لغايه ما ال master يقوم و يشغل ال scheduler و يجى الدور على ال message الى ال slave دا مستنيها فيرد عليه بالداتا الى عايز يبعتها
2. لو ال master مقامش ابعتله request تانى و تالت
3. لو مقامش معد الرابع مينفعش ابعت request رابع غير بعد 1.5 sec

|  |
| --- |
| 1. **lin connections in automotive ECU** |

ال voltage بتاع ال LIN bus هو 12 volt

شغال ازاى ؟

* فى ال receiving
  + 1 > 60% of supply voltage
  + 0 < 40%

عشان دايما فى ال receiving بيبقى فى noise فبيديله margin اكبر

* فى ال transmit
  + 1 > 80%
  + 0 < 20%

عشان احول من logic ال controller (5v) ل logic ال LIN Bus لازم استخدم transceiver

دايما ال transceiver هوا external hardware

