**IPC (internal protocol communication)**

1. Define

The message exchange between imx6 and nxp will have 37 Byte:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | … | Data[31] |
| 4 byte | 1 Byte | 1 Byte | … | 1 Byte |

Message ID: Is the unique identify for each message. This Id is 4 byte range and will auto reset if overflow.

IsResponse: Use to indicate that the message must receive a response. If not, the message will resend until timeout.

* IsResponse = 1, need a response from target.
* IsResponse = 0, Not need a response from target.

Data: The data of message has 32 byte. It depends on the message type.

1. Normal Request Message:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | …. | Data[31] |
| id | 1 | Command Id | 0 | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| Command ID | Expect Action | Expect Response (\*) |
| eDoSilenceCmd | AlarmControl::S\_GetInstance()->DoSilence(); | Accepted/Rejected |
| eCancelSilenceCmd, | AlarmControl::S\_GetInstance()->CancelSilence(); | Accepted/Rejected |
| eManHoldInspKeyDownCmd, | BdTask::S\_GetInstance()->Send(eManHoldInspKeyDown); | Accepted/Rejected |
| eManInspKeyCmd, | BdTask::S\_GetInstance()->Send(eManInspKey); | Accepted/Rejected |
| eManHoldInspKeyUpCmd, | BdTask::S\_GetInstance()->Send(eManHoldInspKeyUp); | Accepted/Rejected |
| eBadFileSystemAlarmActiveCmd, | Alarm[ALARM\_IDX(eBadFileSystem)]->SetIsActive(eActive); | Accepted/Rejected |
| eBadFileSystemAlarmNotActiveCmd, | Alarm[ALARM\_IDX(eBadFileSystem)]->SetIsActive(eNotActive); | Accepted/Rejected |
| eProxyFlowResetLineOnCmd, | ProxyFlowResetLine->SetDesiredState(eOn); | Accepted/Rejected |
| eProxyFlowResetLineOffCmd, | ProxyFlowResetLine->SetDesiredState(eOff); | Accepted/Rejected |
| eImpedanceLineOnCmd, | InpedanceLine->SetDesiredState(eOn); | Accepted/Rejected |
| eImpedanceLineOffCmd, | InpedanceLine->SetDesiredState(eOff); | Accepted/Rejected |
| eSafetyValveOpenCmd, | SafetyValve->SetDesiredState(eOpen); | Accepted/Rejected |
| eExhPressureZeroOnCmd, | ExhPressureZero->SetDesiredState(eOn); | Accepted/Rejected |
| eInhPressureZeroOnCmd, | InhPressureZero->SetDesiredState(eOn); | Accepted/Rejected |
| eExhPressureZeroOffCmd, | ExhPressureZero->SetDesiredState(eOff); | Accepted/Rejected |
| eInhPressureZeroOffCmd, | InhPressureZero->SetDesiredState(eOff); | Accepted/Rejected |
| ePurgeFlowOnCmd, | PurgeFlow->SetDesiredState(eOn); | Accepted/Rejected |
| ePurgeFlowOffCmd, | PurgeFlow->SetDesiredState(eOff); | Accepted/Rejected |
| eOilPumpOnCmd, | OilPump->SetDesiredState(eOn); | Accepted/Rejected |
| eOilPumpOffCmd, | OilPump->SetDesiredState(eOff); | Accepted/Rejected |
| eAirFlowControllerEnableCmd, | AirFlowController-> *Enable* (); | Accepted/Rejected |
| eO2FlowControllerEnableCmd, | O2FlowController->*Enable*(); | Accepted/Rejected |
| eAirFlowControllerDisableCmd, | AirFlowController->*Disable*(); | Accepted/Rejected |
| eO2FlowControllerDisableCmd, | O2FlowController->*Disable*(); | Accepted/Rejected |
| eShutDownRequestKeyCmd, | BdTask::S\_GetInstance()->Send(eShutdownRequestKey); | Accepted/Rejected |
| e100PercentO2Cmd, | BdTask::S\_GetInstance()->Send(e100PercentO2); | Accepted/Rejected |
| e100PercentO2OffCmd, | BdTask::S\_GetInstance()->Send(e100PercentO2Off); | Accepted/Rejected |
| eDoProxyCalibrationCmd, | proxySensor->setCurrentAction(eDoProxyCalibration); | Accepted/Rejected |
| eClearProxyCalibrationCmd, | proxySensor->setCurrentAction(eClearProxyCalibration); | Accepted/Rejected |
| eHummingVOnCmd, | HfoServo::S\_GetInstance()->SetHummingVON(eTrue); | Accepted/Rejected |
| eHummingVOffCmd, | HfoServo::S\_GetInstance()->SetHummingVON(eFalse); | Accepted/Rejected |
| eSVChangeRequestCmd, | BdTask::S\_GetInstance()->Send(eSVChangeRequest); | Accepted/Rejected |
| eAmpChangeRequestCmd, | BdTask::S\_GetInstance()->Send(eAmpChangeRequest); | Accepted/Rejected |
| eFreqChangeRequestCmd, | BdTask::S\_GetInstance()->Send(eFreqChangeRequest); | Accepted/Rejected |
| eMapChangeRequestCmd, | BdTask::S\_GetInstance()->Send(eMapChangeRequest); | Accepted/Rejected |
| eSettingChangeAlarmActiveCmd, | Alarm[ALARM\_IDX(eSettingChange)]->SetIsActive(eActive) ; | Accepted/Rejected |
| eSettingChangeAlarmNotActiveCmd, | Alarm[ALARM\_IDX(eSettingChange)]->SetIsActive(eNotActive) ; | Accepted/Rejected |
| eGetProxyVersionCmd, | proxySensor->setCurrentAction(eGetProxyVersion); | Accepted/Rejected |
| eCMVServoControllerEnableCmd, | CMVServoController->*Enable*(); | Accepted/Rejected |
| eCMVServoControllerDisableCmd, | CMVServoController->*Disable*(); | Accepted/Rejected |
| eAlarmResetKeyCmd, | BdTask::S\_GetInstance()->Send(eAlarmResetKey); | Accepted/Rejected |
| eProxySensorGetOffsetCmd, | proxySensor->getOffset(); | Accepted/Rejected |

(\*) Accepted – if the expect action is done successfully and vice versa is Rejected

1. Motor Set Baud Rate Message

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | Data[2-5] | …. | Data[31] |
| id | 1 | eSetBaudRateCmd | Motor id | Speed | 0 | 0 |

Motor id: Indicate the motor will be set baud rate.

|  |  |
| --- | --- |
| Motor ID | Motor pointer (Device/Stepmotor.h) |
| eO2ValveId | O2Valve |
| eAirValveId | AirValve |
| eExhValveId | ExhValve |
| eHfoValveId | HfoValve |

Data[2] = LSB of Speed

Data[5] = MSB of Speed

Expect action: (motor pointer)->setBaudRate(Speed);

Expect response: Accepted/Rejected.

1. Motor Configure Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | Data[2] |
| id | 1 | eMotorConfigureCmd | Motor id | size |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data[3] | … | Data[size] | Data[size + 1] | … | Data[31] |
| DataSend | | | 0 | 0 | 0 |

DataSend: An array of byte which is send to motor.

Expect action:

(motor pointer)-> dataToSend = DataSend;

(motor pointer)-> SendtoMotor();

Expect response: Accepted/Rejected.

1. Motor Event Data Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | Data[2] | Data[3-4] | …. | Data[31] |
| id | 1 | eMotorEventDataCmd | M\_ID | M\_Type | M\_Data | 0 | 0 |

Expect action:

MotorEventData data;

data.M\_ID = M\_ID;

data.M\_Type = M\_Type;

data.M\_Data = M\_Data;

DiagnosticPhase::S\_GetInstance()->Send(data);

Expect response: Accepted/Rejected.

1. Diagnotics Command Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | Data[2] | Data[3] | …. | Data[31] |
| id | 1 | eDianogticsCmd | ID | Cmd | Stt | 0 | 0 |

Expect action:

CommandData cmd;

cmd.ID = ID;

cmd.Cmd = Cmd;

cmd.Stt = Stt;

DiagnosticPhase::S\_GetInstance()->Send(cmd);

Expect response: Accepted/Rejected.

1. Single Setting Message

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | Data[2-5] | …. | Data[31] |
| id | 1 | eSettingCmd | ID | Value | 0 | 0 |

Expect action:

BdSingleSettingChangeEvent E;

E.Id = ID;

E.Value = Value;

BdTask::S\_GetInstance()->Send(E);

Expect response: Accepted/Rejected.

1. Normal Response Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | …. | Data[31] |
| id | 0 | Value | 0 | 0 |

Value: 0 is Accepted. 1 is Rejected.

1. Request Data From Motor Message

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | …. | Data[31] |
| id | 1 | eRequestDataFromMotorCmd | Motor id | 0 | 0 |

Expect action: QByteArray dataFromMotor = (motor pointer)->RecievefromMotor();

Expect response:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | Data[2] | …. | Data[31] |
| id | 0 | eRequestDataFromMotorCmd | dataFromMotor | | | |

1. Request Current Mode ID Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | …. | Data[31] |
| id | 1 | eGetCurrentModeIdCmd | 0 | 0 |

Expect action: E\_VentilationMode currentMode = ModeMgr::S\_GetInstance()->GetCurrentModeId();

Expect response:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | …. | Data[31] |
| id | 0 | eGetCurrentModeIdCmd | currentMode | 0 | 0 |

1. Request Proxy Last Status Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | …. | Data[31] |
| id | 1 | eGetProxyLastStatusCmd | 0 | 0 |

Expect action: ProxyStatus status = proxySensor->getLastStatus();

Expect response:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | …. | Data[31] |
| id | 0 | eGetProxyLastStatusCmd | status | 0 | 0 |

1. Request Humming V Status Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | …. | Data[31] |
| id | 1 | eGetHummingVOnCmd | 0 | 0 |

Expect action: E\_Bool status = HfoServo::S\_GetInstance()->GetHummingVON();

Expect response:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Message ID | IsResponse | Data[0] | Data[1] | …. | Data[31] |
| id | 0 | eGetHummingVOnCmd | status | 0 | 0 |