IEEE 1722 Control Format
proposals for new I2C and SPI ACF types

2020-05-26

Edited by Yong Kim
yongkim at axonne dot com

Motivation

- One of the major justifications of automotive Multi-gig Ethernet PHY (IEEE 802.3ch) is network connection of high bandwidth autonomous drive (AD)/advanced driver assistance system (ADAS) sensors, such as uncompressed AD and parking cameras.
- Sensors has control interfaces such as I2C, SPI, and other popular I/F.
- There is a need to transport these control interfaces over a standard L2/L3 transport; ideally, transport protocol that supports time sensitive networking, i.e. IEEE 1722 AVTP.
- Avoid pre-standard and proprietary implementation islands that form once system productions start; Avoid artificial turf defense when these form. There is no intrinsic value to having multiple different but functionally equivalent approaches.
- Stated goal of 1722b – serve the industry and perform necessary revision quickly – fits well with a goal of this proposal.

I2C (Inter-Integrated Circuit) Bus

- Brief Intro to I2C
  - Original Design – Philips Semiconductor (now NXP).
  - True two wire bus, master-slave, master arbitration (CSMA/CA) defined but not in common use.
  - Relevant design branches: SMBus (System Management Bus, Intel), SCCB (Omivision).
  - For more info, refer to available documents/specs online. BTW, wiki entry is not bad at all.
- Real-time Bus Signaling Particulars
  - Single byte read or write transfers
  - Each transfer with acknowledgement – ack bit (9^{th} bit).
  - Read cycle allows for clock stretching (open drain clock) for slave to get the data ready before actual data read cycle to begin.
- Parallels to CAN bus
  - CAN bus has the same bus data bit transfer ack signal (I2C: ack bit; CAN: error-status-indicator (esi)), AND master read of slave device data (I2C: read; CAN: remote-transmission-request (rtr)).
  - I2C messages are normally asynchronous but could also be implicitly synchronous. Maps well to both CAN and CAN_brief ACF types.
  - CAN identifier is multi-purpose, but in context of device address, it maps well to I2C in-band addressing.
  - NOT: CAN id implicitly defines the payload contents and data byte lengths. I2C is almost always byte transfer at a time.

SPI (Serial Peripheral Interface) Bus

- Brief Intro to SPI
  - Original Design – Motorola Semiconductor (➔ Freescale, now NXP).
  - Four wire bus, full-duplex, master to multiple slaves, each additional slave gets its own CS.
  - Relevant design branches: SDIO (Secure digital), double and quad data lines (not same as QSPI), etc.
  - For more info, refer to available documents/specs online. BTW, wiki entry is not bad at all.
- Real-time Bus Signaling Particulars
  - No explicit address in-band (CS instead). Any number of bytes in transfer – most up to 4K (matches storage block)
  - Polarity of Data (active high/low) and Clock (falling or rising) – 4 different modes.
  - Wild-west of protocol definitions due to the lack of formal specification nor associated standards body.
    Queued SPI (QSPI), double datarate SPI, eSPI, dual and quad SPI, etc.
    But emerging automotive relevant SafeSPI – protocol and payload, not bus behavior is interesting (and in use).
- Parallels to Lin bus
  - NOT: Lin is simplex. SPI is duplex.
  - Lin uses no hardware flow control, SPI does not have hardware flow control.
  - Lin messages are normally synchronous (master always provide clock) and who knows whether this aspect of synchronous behavior is actually used or not (few ways to find out). SPI is always synchronous to master clock.
  - NOT: Lin identifier is in context of device address could map to SPI CS, but not very relevant.
  - NOT: CAN id implicitly defines the payload contents and data byte lengths. I2C is almost always byte transfer at a time.

I2C ACF Type proposal

Fields	CAN	I2C	Notes
acf_msg_type	0x01 or 0x02	0x0D or 0xOE	0001 → 1101; 0010 → 1110
acf_msg_length	0x02 ~ 05, or more (FD)	0x03	I2C always a byte
mtv (msg ts valid)	as-is	same	Timestamp valid. False for _brief
rtr (rmt_tx_req)	as-is	read ack (9th bit)	I2C stateful, as CAN is for a read cycle.
eff (ext_frame_fmt)	as-is	re-use/spare	I2C – spare/reserved.
brs (bit-rate_switch)	as-is	re-use Master STOP	I2C Master STOP bus cycle
fdf (flex d rate fmt)	as-is	re-use Master START	I2C Master START bus cycle
esi (err state indct)	as-is	ack (9th bit)	I2C stateful, as CAN is when esi is used.
can_bus_id [5]	as-is	same	I2C (network) bus ID
msg_ts (not in _brief)	as-is	same	
can_id [29]	as-is	Same and reduced [8]	I2C uses 7 bit (8 bit rare)
can_msg_payload	as-is	Same and reduced	I2C Fixed to 1 byte (so 1 quadlet).

SPI ACF Type proposal

Fields	LIN	SPI	Notes
acf_msg_type	0x03	0x0F or 0xOE	0011 ➔ 1111;
acf_msg_length	0x03 or 0x04	variable	LIN: 2~8 bytes; SPI:1~4K+ bytes SafeSPI: 32 bits.
mtv (msg ts valid)	as-is	same	Timestamp valid. False for _brief
lin_bus_id [5]	as-is	same	SPI (network) bus ID
msg_ts	as-is	same	
lin_id [8]	as-is	same and may reduced	SPI maps to respective HW slave CS. Up to 10 bits bot not required. SafeSPI T(S)A min 3 (of 10) bits.
lin_msg_payload	as-is	same and expanded	SPI variable MTU to 512, and 4K. Note: discuss whether there is a need for byte-wise (for example, control), or streaming queued transport (for example, QSPI, SDIO).
spi_msg_queued	N/A	QSPI	Queued SPI operation (TBD discussion whether to define this – spec needed)