Controller Area Network with Flexible Data for LPC5461x

presented by

Andy Ayre
Embedded Systems Academy, Inc.

www.esacademy.com
Webinar Contents

- LPC family introduction
- Introduction to ESA
- Differences between CAN and CAN-FD
- CAN-FD Protocol Overview
- CAN-FD on the LPC5461x
- CAN Controller Block Diagram
- CAN-FD Applications
- CAN-FD Support
- CANopen Stack Availability
- Software API
- API Demonstration
Why Developers Select LPC Microcontrollers

Investing in Innovative & Differentiated Technologies to Maintain our Global Leadership in the Broad Market
LPC546xx Block Diagram

**CPU**
- 180MHz Cortex-M4F

**Memory**
- Up to 512 KB Flash, Up to 200 KB RAM
- 16 KB EEPROM

**Interfaces for connectivity & sensors**
- Stereo DMIC subsystem
- 1x HS USB (H/D) w/ on-chip HS PHY, XTAL-less FS USB (H/D)
- 10 SPI, 10 I2C, 10 UART, 2 I2S channels (max 10 channels total)
- Graphic LCD with resolutions up to 1024x768
- Ethernet with IEEE1722 timestamp
- 2 x CAN-FD controller
- Quad SPI flash interface
- External Memory interface (up to 32 bits)

**Packages**
- LQFP208 (28 x 28 mm), TFBGA180 (12 x 12 mm)
- LQFP100, TFBGA100

**Operating**
- Operating voltage: 1.71 to 3.6V
- Temperature range: -40 to 105 °C
LPCXpresso54618 CAN-FD Kit

Development platform for LPC546xx Series

- LPC54618 MCU running at 180MHz
- 128Mb Micron SDRAM
- 128Mb Micron quad SPI flash
- Built-in CMSIS-DAP/J-link debug probe
- Ethernet, DMIC, SD card, USB HS/FS ports
- Stereo audio codec
- Arduino UNO R3 compatible expansion ports
- Shield board with TJA1059 dual transceiver
- Supported by MCUXpresso SDK for MCUXpresso IDE, Keil and IAR tools
Embedded Systems Academy, Inc.

- Founded 1999
- Consulting services
- Training services
- CANopen stacks, PC development/analysis tools, bootloaders
- Participate in CANopen standardization
- J1939
- Flash Magic
DIFFERENCES BETWEEN CAN AND CAN-FD
Differences between CAN and CAN-FD

- Invented by Bosch in 2011
- Finalized and ISO standardized 2015
  - ISO 11898-1:2015
- Significantly higher bitrates for data
  - Limited by transceivers in practice
- More data per frame
  - Up to 64 bytes
- Up to six times higher throughput
- Can mix FD and non-FD CAN controllers
- Can mix FD and non-FD frames
- Need FD-compliant transceivers above 1Mbps
Differences between CAN and CAN-FD

- Remote transmission requests not supported
- Bit to indicate transmitter error status
- Improved error checking
- Bus topology and wiring stays the same
  - Same networking costs
CAN-FD PROTOCOL OVERVIEW
## CAN-FD Protocol Overview

- **Familiar frame format**

<table>
<thead>
<tr>
<th>SOF</th>
<th>Arbitration field</th>
<th>Control field</th>
<th>Data field (payload)</th>
<th>CRC field</th>
<th>ACK field</th>
<th>EOF</th>
<th>IMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>12 or 32* bit</td>
<td>8 or 9* bit</td>
<td>0 to 64* byte</td>
<td>28 or 33 bit*</td>
<td>2 bit</td>
<td>7 bit</td>
<td>3 bit</td>
</tr>
</tbody>
</table>

  **MSB**

  **LSB**

Diagrams © CiA
CAN-FD Protocol Overview

- Arbitration field 11-bit identifier
- Arbitration field 29-bit identifier
CAN-FD Protocol Overview

- CAN-FD Control field

<table>
<thead>
<tr>
<th>No. ofting bytes</th>
<th>Data length code (DLC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DLC3</td>
</tr>
<tr>
<td>0 to 8</td>
<td>As in Classical CAN</td>
</tr>
<tr>
<td>12</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
</tr>
<tr>
<td>32</td>
<td>r</td>
</tr>
<tr>
<td>48</td>
<td>r</td>
</tr>
<tr>
<td>64</td>
<td>r</td>
</tr>
</tbody>
</table>

Diagrams © CiA
CAN-FD Protocol Overview

- CAN-FD Checksum field

![Diagram showing CAN-FD Checksum field](image-url)
CAN-FD Protocol Overview

- “Classical” and “FD” frame formats
- FDF - Uses previously reserved bit to select FD
- BRS - bit used to change data bitrate
CAN-FD Protocol Overview

- Scope waveform
# CAN-FD Protocol Overview

<table>
<thead>
<tr>
<th></th>
<th>11-bit ID</th>
<th>29-bit ID</th>
<th>FDF</th>
<th>BRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-FD 11-bit</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-FD 29-bit</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD 11-bit</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FD 11-bit, fast data</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FD 29-bit</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FD 29-bit, fast data</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
CAN-FD Protocol Overview

- Independent sample points
- ESI - bit to indicate error status of transmitter
- Frame includes encoding of number of stuff bits
CAN-FD ON THE LPC5461X
CAN-FD on the LPC5461x

- Two CAN controllers
- Conforms to ISO11898-1
  - Supports CAN-FD ISO mode
  - Supports all CAN-FD functionality
- Global FD and bit rate switching enable
- Flexible RAM configuration
- RAM sharing between CAN controllers
  - Ideal for bridging applications
### CAN-FD on the LPC5461x

The diagram illustrates the memory mapping and buffer sizes for CAN-FD on the LPC5461x. Here is the breakdown:

<table>
<thead>
<tr>
<th>Memory Segment</th>
<th>Start Address</th>
<th>Description</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIDFC.FLSSA</td>
<td>0-128 elements / 0-128 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XIDFC.FLESA</td>
<td>0-64 elements / 0-128 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXF0C.F0SA</td>
<td>0-64 elements / 0-1152 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rx FIFO 0</td>
<td>0-64 elements / 0-1152 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rx FIFO 1</td>
<td>0-64 elements / 0-1152 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rx Buffers</td>
<td>0-64 elements / 0-1152 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx Event FIFO</td>
<td>0-32 elements / 0-64 words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx Buffers</td>
<td>0-32 elements / 0-576 words</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The memory mapping and buffer sizes are specific to the LPC5461x microcontroller.
CAN-FD on the LPC5461x

Receive

- Two receive FIFOs
  - Circular or non-circular
  - Configurable fill-level interrupt
  - Configurable size

- 64 dedicated receive buffers
- Flexible acceptance filtering
- Global simplified acceptance filtering
- Timestamping
- High priority messages
CAN-FD on the LPC5461x

- Transmit FIFO/queue
  - Configurable size
- Up to 32 Transmit buffers
CAN-FD on the LPC5461x

Transmit

- Transmit events
  - Stored in a FIFO
  - Configurable FIFO size
  - Timestamps
CAN-FD on the LPC5461x

- Dual interrupts per CAN controller
  - Configurable

- AUTOSAR support
  - Can be used with other higher layer protocols, e.g. CANopen, J1939

- Test modes
  - Loopback
  - Pin monitoring
  - Direct pin driving

- Restricted operation
  - Autobauding

- Power down
CAN-FD on the LPC5461x

- Transmitter delay compensation
CAN CONTROLLER BLOCK DIAGRAM
LPC5461x CAN Block Diagram

- CAN Core
- Interrupt IF (int0 & int1)
- M_CAN sub_sys
- AHB Slave IF
- AHB Master IF
- Memory IF
- Clk
- Sync
- Tx_State
- Tx_Req
- Rx_State
- Tx Handler
- Rx Handler
- Tx Prioritization
- Acceptance Filter

CAN Clock Domain
Host Clock Domain
CAN-FD APPLICATIONS
CAN-FD Applications

- Automotive, Medical
  - Improved error detection
- Firmware updates
  - Faster data transfers, lower overhead
- Display data
  - Better suitability for transferring periodic blocks of data
- Security
- Data logging
CAN-FD Support

- **Embedded Systems Academy, Inc.**
  - Provides full service support
  - CAN and CAN-FD
  - NXP forums
  - Consulting

- CANopen Stack
- CANopen development tools
SOFTWARE API
Software API

- Based on the FLEXCAN API for Kinetis K60
- Example
  - Uses ISO CAN-FD, 64 bytes per frame
  - Uses 1Mbps nominal, 4Mbps data
  - Demonstrates transmit and receive
- Setup
  - NXP LPC5461x evaluation board
  - NXP CAN-FD shield
  - PC CAN-FD interface, e.g. PEAK PCAN-USB FD
  - Regular 120 Ohm terminated cable
Software API
Configuration

- **Board_CAN0_Init()**
  - Sets up pin multiplexing

- **CAN_GetDefaultConfig()**
  - Gets a default configuration by filling in a structure
  - 125kbps nominal and data
  - 0x2001000 RAM base address
  - No RTRs
  - FD enabled

- **CAN_Init()**
  - Uses a configuration structure to initialize CAN controller

- **CAN_Enable()**
  - Enables CAN controller
Software API Configuration

- `can_config_t`
  - `nominalBaudRate`
  - `dataBaudRate`
  - `baseAddress`
  - `timestampClock_Hz`
  - `rejectStandardRTR / rejectExtendedRTR`
  - `enableLoopBack`
  - `enableNonISOMode`
  - `disableFD`
Software API
Receive Configuration

- **CAN_SetRxFifoConfig()**
  - Sets the size of a receive FIFO and enables the FIFO
  - Supports both FIFOs

- **CAN_SetRxGlobalMask()**
  - Sets the global receive settings to accept all 11-bit or 29-bit messages into a FIFO

- **CAN_SetRxIndividualMask()**
  - Configures which 11-bit messages can be received into a FIFO or a message buffer
  - Selection of destination is by using a macro
    - FLEXCAN_RX_FIFO0_STD_MASK
    - FLEXCAN_RX_FIFO1_STD_MASK
    - FLEXCAN_RX_MB_STD
software API
Receive Configuration

- CAN_SetRxExtIndividualMask()
  - Configures which 29-bit messages can be received into a FIFO or a message buffer
  - Selection of destination is by using a macro
    - FLEXCAN_RX_FIFO0_EXT_MASK_LOW / HIGH
    - FLEXCAN_RX_FIFO1_EXT_MASK_LOW / HIGH
    - FLEXCAN_RX_MB_EXT_LOW / HIGH
Software API
Transmit configuration

- **CAN_SetTxMbConfig()**
  - Enables a transmit message buffer

- **CAN_TransferCreateHandle()**
  - Creates a “handle” for a specific CAN controller
  - One handle per CAN controller
  - Associates a CAN controller with a callback function
Software API
Transmit – Message Buffers

- **CAN_TRANSFERSENDBlocking()**
  - Transmits a CAN message and waits for it to go out onto the bus
  - Uses `flexcan_frame_t`
    - length
    - type
    - format
    - proto
    - bitratemode
    - id
    - dataWord / dataByte
Software API
Transmit – Message Buffers

- **CAN_TransferSendNonBlocking()**
  - Transmits a message and returns immediately
  - Passed is a handle and the message
  - Callback function called on transmission
    - handle and status

- **CAN_TransferAbortSend()**
  - Aborts send
Software API
Transmit – FIFO/Queue

- **CAN_WriteTxFIFO()**
  - Puts a message into the FIFO/queue
  - Returns immediately
Software API
Receive

- **CAN_ReadRxFIFO()**
  - Polls a FIFO
  - Fills in structure if message waiting to be read

- **CAN_ReadRxMb()**
  - Polls a message buffer
  - Fills in structure if message waiting to be read
Software API
Receive

- **CAN_TransferCreateHandle()**
  - Creates a “handle” for a specific CAN controller
  - One handle per CAN controller
  - Associates a CAN controller with a callback function

- **CAN_TransferReceiveNonBlocking()**
  - Returns immediately
  - Passed is a handle and rx structure
    - Message structure
    - Message buffer number
  - Callback function called on reception
    - handle and status

- **CAN_TransferAbortReceive()**
  - Aborts message buffer receive
Software API

Receive

- **CAN_TransferReceiveFifoNonBlocking()**
  - Non-blocking receive for FIFOs

- **CAN_TransferAbortReceiveFifo()**
  - Aborts FIFO receive
Software API

Miscellaneous

- CAN_DelInit()
- CAN_GetBusErrCount()
- CAN_GetStatusFlags()
SOFTWARE API DEMONSTRATION
Where to get started

- LPC range of MCUs at nxp.com/lpc
- LPCXpresso54618 board at nxp.com/demoboard/om13094
- CAN-FD driver add-ons under Downloads tab
- Free tools and software at nxp.com/mcuxpresso