CAN Technical Overview
Agenda

- Overview of CAN
  - What is CAN?
  - Why CAN?
- CAN Protocol:
  - CAN 2.0A & CAN 2.0B
  - Basics Concepts & Definitions
  - Identifiers & Arbitration
  - Robustness & Flexibility
  - Message Formats
  - Errors at Message and Bit Level
  - Error Handling and Confinement
- CAN Implementations
  - The Requirements of a CAN Controller
  - Full CAN vs. Basic CAN Controllers
  - Message Buffering & Filtering
  - Effective, Low Cost CAN Receive Structures
- Motorola CAN Modules
- Summary

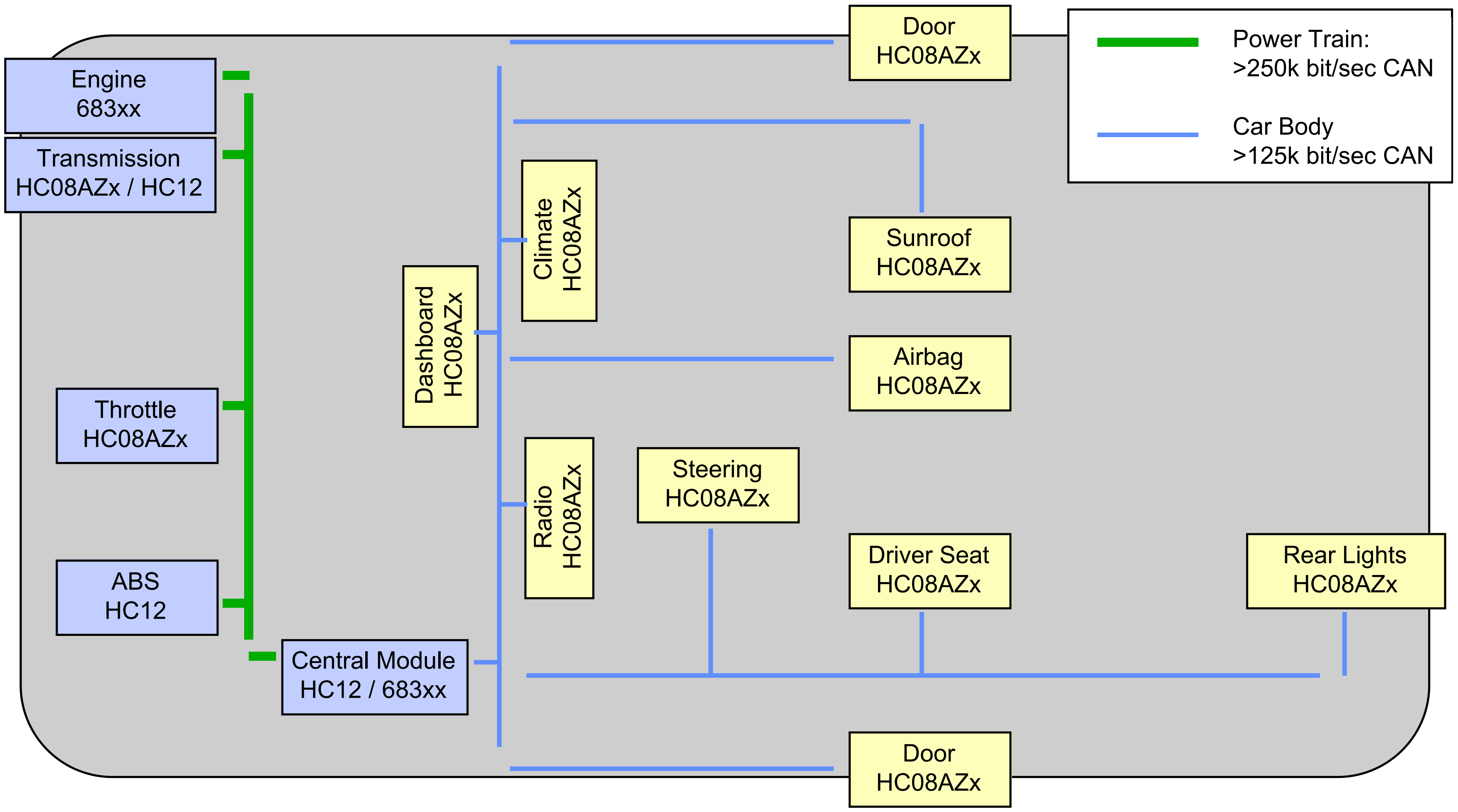
Overview of CAN

“What is CAN?!”

• CAN (Controller Area Network) is a multiplexed serial communication channel which data is transferred among distributed electronic module; very similar to SPI or SCI, although more complex.

-MultiMaster, MultiCast Protocol without Routing.

Typical CAN Network



CAN 2.0
Structure of a CAN Node

- ISO 7498 defines the communication's standard Open Systems Interconnection (OSI).
  - This standard defines seven independent layers of a protocol stack.

- CAN Specification, ISO 11898, deals with only the Physical & Data Link Layers for a CAN network.

Layer 7:	Application Layer
Layer 6:	Presentation Layer
Layer 5:	Session Layer
Layer 4:	Transport Layer
Layer 3:	Network Layer
Layer 2: Data Link Layer	**Logic Link Control** Data Transfer Remote Data Request Message Filtering Recovery Management & Overload Notification
	Medium Access Control Framing & Arbitration Error Checking & Error Flags Fault Confinement Bit Timing
Layer 1:	Physical Layer

CAN 2.0A vs CAN 2.0B

CAN 2.0A:

11 Bit Identifier

M68HC05X Family

- Used by vast majority of current applications.
- **Greater message throughput** and **improved latency times**
- ***Less silicon overhead !***

CAN 2.0B

29 Bit Identifier

HC08 / HC11 + MSCAN

- Originally defined for USA Passenger Cars but now their Taskforce decree that it is **not necessary**.
- Allows more information in message but requires **more bus bandwidth**
- ***More silicon cost and less efficient use of bus !***

Team Automotive

MOTOROLA

CAN vs Other Protocols

	CAN 2.0A/B	SAE J1850	BEAN
Bit Encoding	NRZ	PWM or VPW	NRZ
Bus Wire Medium	Single or Dual	Single (10.4Kbps) or Dual (41.0Kbps)	Single
Data Rate	1Mbps	10.4 Kbps VPW or 41.7 Kbps PWM	10kbps
# of SOF Bits	1bit	unique symbol	1 bit
# of Identifier Bits	11/29 bits	8 to 24 bits	12 bits
Data Length Code	4 bits	none	4 bits
Message Length Field	0 to 24 bits	0 to 24 bits	1 to 88
CRC Field	15 bits	8 bits	8 bits
ACK Field	2 bits	none	2 bits
End of Frame	7 bits	unique symbol	6 bits
IFR	ACK	a) 1 byte from 1 receiver or- b) Multiple bytes from multiple receivers. c) Data bytes, with or without CRC, from a single receiver.	
EOF	1 bit	1 bit	1 bit

“Why CAN?!”

- Widely Accepted Standard
  - “THE STANDARD” in the automotive industry in Europe.
  - Gaining acceptance in the US.
- Robust
  - Handles extreme conditions well.
  - Simple to configure.
  - Good error detection capabilities.
  - Excellent two-wire fault tolerance capabilities:
    - » Either of the two wires in the bus is broken.
    - » Either of the two wires are shorted to power, to ground, or together.
- ‘Lots of software and hardware support available
  - Application layer and driver software available.
  - CAN Bus Analyzer/development tools.
  - CAN USER group conferences.

Technical Overview

Identifiers
Arbitration
CAN 2.0A & 2.0B Message Frames
Network Flexibility & Expanision
Resynchronization

Identifiers

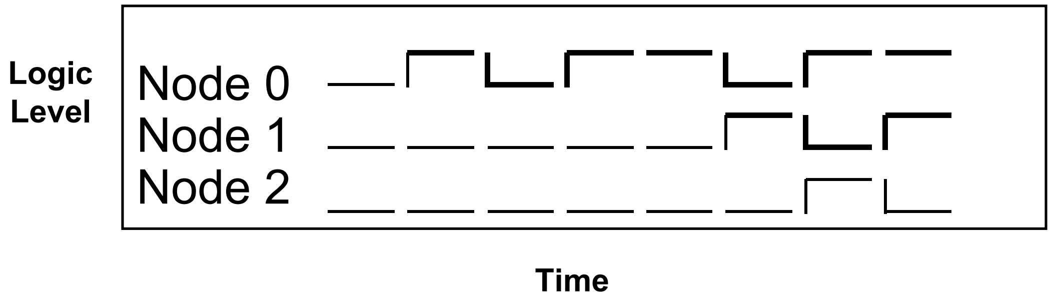
- Labels the content (type) of a message.
- Performs acceptance test of messages.
- Arbitrates & determines the priority of the message.

Arbitration

- Carrier Sense, Multiple Access with Collision Detect (CSMA/CD)
  - Method used to arbitrate and determine the priority of messages.
  - Uses enhanced capability of non-destructive bitwise arbitration to provide collision resolution.

Bitwise Arbitration

- Any potential bus conflicts are resolved by bitwise arbitration
  - Dominant state (logic 0) has precedence over a recessive state (logic 1).



» Competition for the bus is won by node 2 .

» Nodes 0 and 1 automatically become receivers of the message

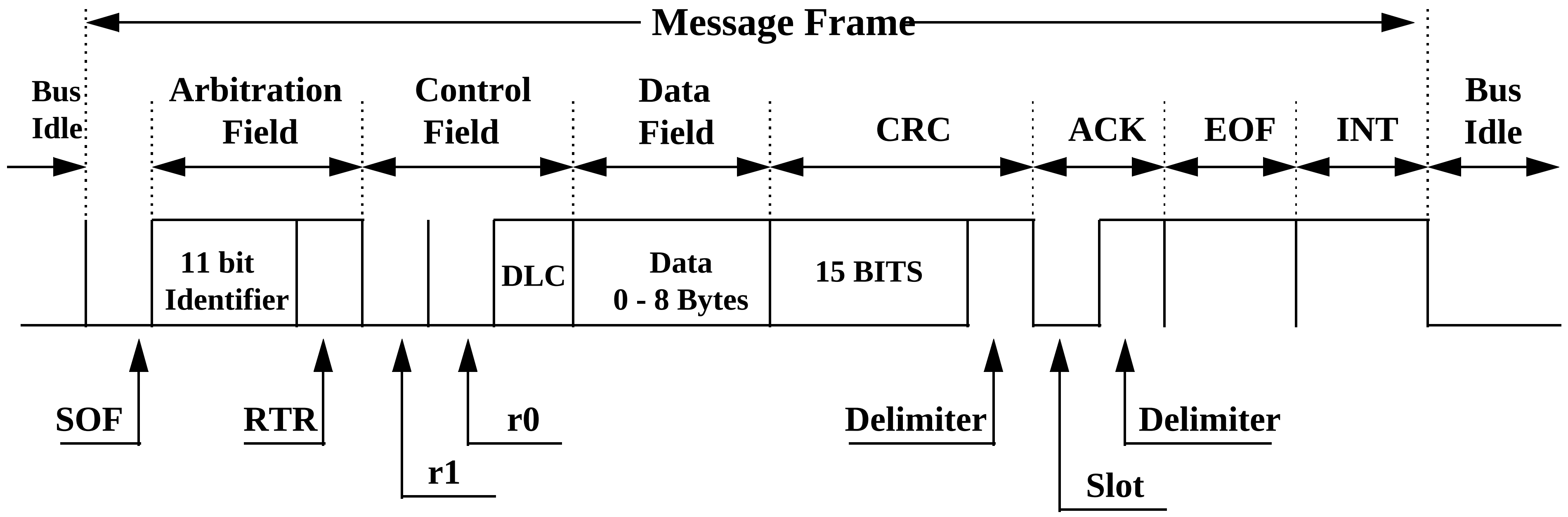
» Nodes 0 and 1 will re-transmit their messages when the bus becomes available again.

Local Priority Concept

- Each transmit buffer hold an 8-bit priority register.
- Allows flexible priority schemes.
  - Fixed Priority for each buffer.
  - Map a CAN ID to a priority.
  - First in, first out.
  - Back - to - Back transmission of same ID.
  - Situation priority.

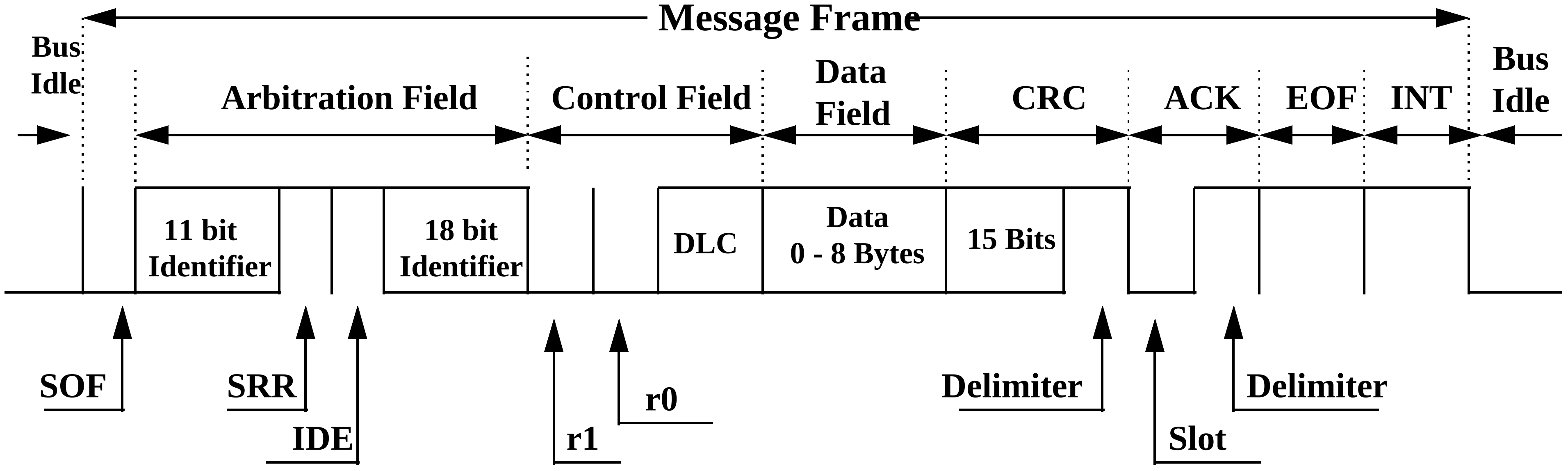
CAN 2.0A Message Frame

- CAN 2.0A (Standard Format)
  - 11 bit message identifier
  - Transmits and receives only standard format messages.



CAN 2.0B Message Frame

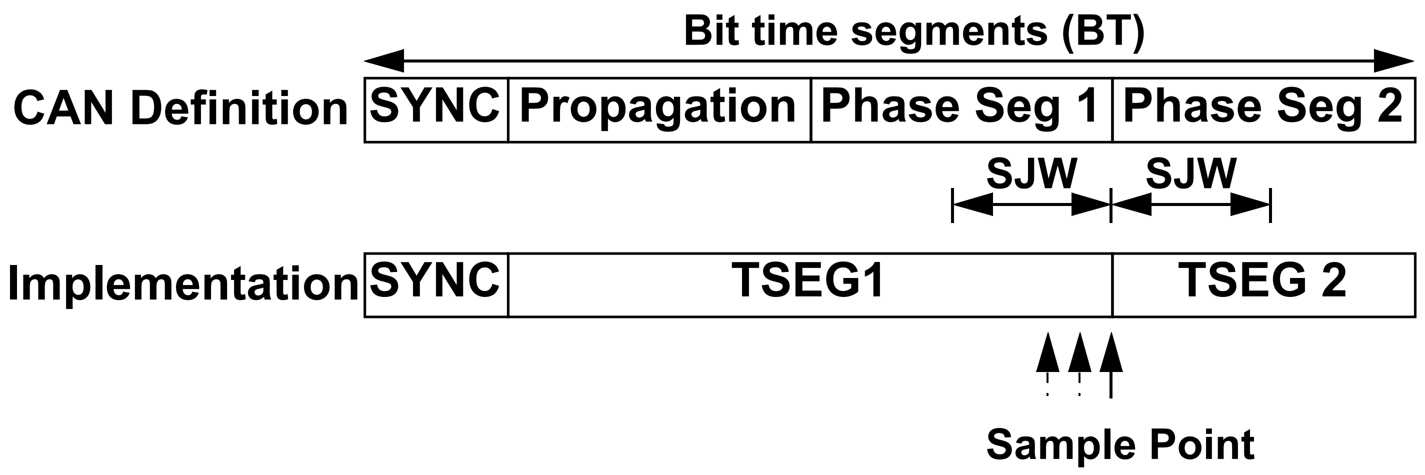
- **CAN 2.0B (Extended Format)**
  - **- Capable of receiving CAN 2.0A messages.**
  - **- 29 bit message identifier. 11 bits for a CAN 2.0A message + 18 bits for a CAN 2.0B message.**



Network Flexibility and Expansion

- High degree of flexibility for system configuration.
  - Easy to add new (purely) receiving nodes.
  - Measurements needed by several controllers can be transmitted via the bus.

CAN Resynchronization



•Sync-Seg : Used to synchronize the nodes on the bus and the start of bit transition.

•Prop-Seg : A period of time that is used to compensate for physical delay times within the network.

•Phase-Seg1: A buffer segment that may be lengthened during resynchronization to compensate for oscillator drift and positive phase differences between the oscillators of the transmitting and receiving node(s).

•Phase-Seg2: A buffer segment that may be shortened during resynchronization to compensate for negative phase errors and oscillator drift.

•SJW : 1bit time, synchronization jump width (SJW) is not to be exceeded.

Technical Overview

Five Error Detection Mechanisms

Error Flags, Confinement, & Counts

Error Active Mode

Bus-Off Mode

Error Levels

Overload Frame

Error Detection

- CAN implements five error detection mechanisms.
  - Three at the message level
    - » Cyclic Redundancy Checks (CRC)
    - » Frame Checks
    - » Acknowledgment Error Checks
  - Two at the bit level
    - » Bit Monitoring
    - » Bit Stuffing

MOTOROLA

Cyclic Redundancy Checks (CRC)

•CRC Errors (Message Level)

–The 15 bit CRC is computed by the transmitter and based on the message content.

– All receivers that accept the message, recalculates the CRC and compares against received CRC.

– If the two values do not match a CRC error is flagged.

Frame Check
Message Level

- If a receiver detects an invalid bit in one of these positions, a Form Error (or Format Error) will be flagged:
  - CRC Delimiter
  - ACK Delimiter
  - End of Frame Bit Field
  - Interframe Space (the 3 bit INTermission field and a possible Bus Idle time).

ACK Error Check
Message Level

- Each receiving node writes a dominant bit into the ACK slot
- If a transmitter determines that a message has not been ACKnowledged then an ACK Error is flagged.
- ACK errors may occur because of transmission errors because the ACK field has been corrupted or there is no operational receivers.

MOTOROLA

Bit Monitoring
Bit Level

- Each bit level (dominant or recessive) on the bus is monitored by the transmitting node.
  - Bit monitoring is not performed during arbitration or on the ACK Slot.

Bit Stuffing
Bit Level

- Bit stuffing is used to guarantee enough edges in the NRZ bit stream to maintain synchronization.
  - After five identical and consecutive bit levels have been transmitted, the transmitter will automatically inject (stuff) a bit of the opposite polarity into the bit stream.
  - Receivers of the message will automatically delete (de-stuff) such bits.
  - If any node detects six consecutive bits of the same level, a stuff error is flagged.

Error Flag

- If an error is detected by at least one node
  - The node that detects the error will immediately abort the transmission by sending an Error Flag.
- An Error Flag consists of six dominant bits.
  - This violates the bit stuffing rule and all other nodes respond by also transmitting Error Flags.

Error Confinement

- A method for discriminating between temporary errors and permanent failures .
  - Temporary errors may be caused by external conditions, voltage spikes, etc.
  - Permanent failures may be caused by bad connections, faulty cables, defective transmitters or receivers, or long lasting external disturbances.

MOTOROLA

Error Counts

- The error flags are counted and stored.
  - Receive errors are given a weighting of 1.
  - Transmit errors are given a weighting of 8.
- Transmitting node errors can be quickly detected.

Error Active Mode

- If an error count in either the transmit or receive register is greater than zero, the node enters Error Active mode.
  - Error Active nodes are still fully functional, but are in an alert condition.
  - Subsequent good messages decrement the Error Count registers by a count of 1.
  - If no further errors are detected, and both Error Counts go back to zero, an Error Active node returns to Normal mode.

Bus-Off Mode

- If the Error Count in either the transmit or receive registers exceeds 255, the node will take itself off-line by going into “Bus-Off mode”.
  - Permanently faulty nodes will cease to be active on the bus.
- Nodes in the Bus Off state can go back to Error Active mode
  - 128 occurrences of 11 consecutive recessive bits on the bus need to occur.

Error Flag Levels (MSCAN HC08/HC12)

- The alarm levels are defined by CAN 2.0

- MSCAN implements a level sensitive flag for each error condition

- Error Active Mode..............................	001 to 127
- Warning...	096
- Error Passive Mode...........................	128 to 254
- BusOff Mode......................................	> 255

MOTOROLA

Overload Frame

- If a CAN node receives messages faster than it can process the messages, then the CAN module will signal an overload condition and then send an overload interrupt to the CPU.

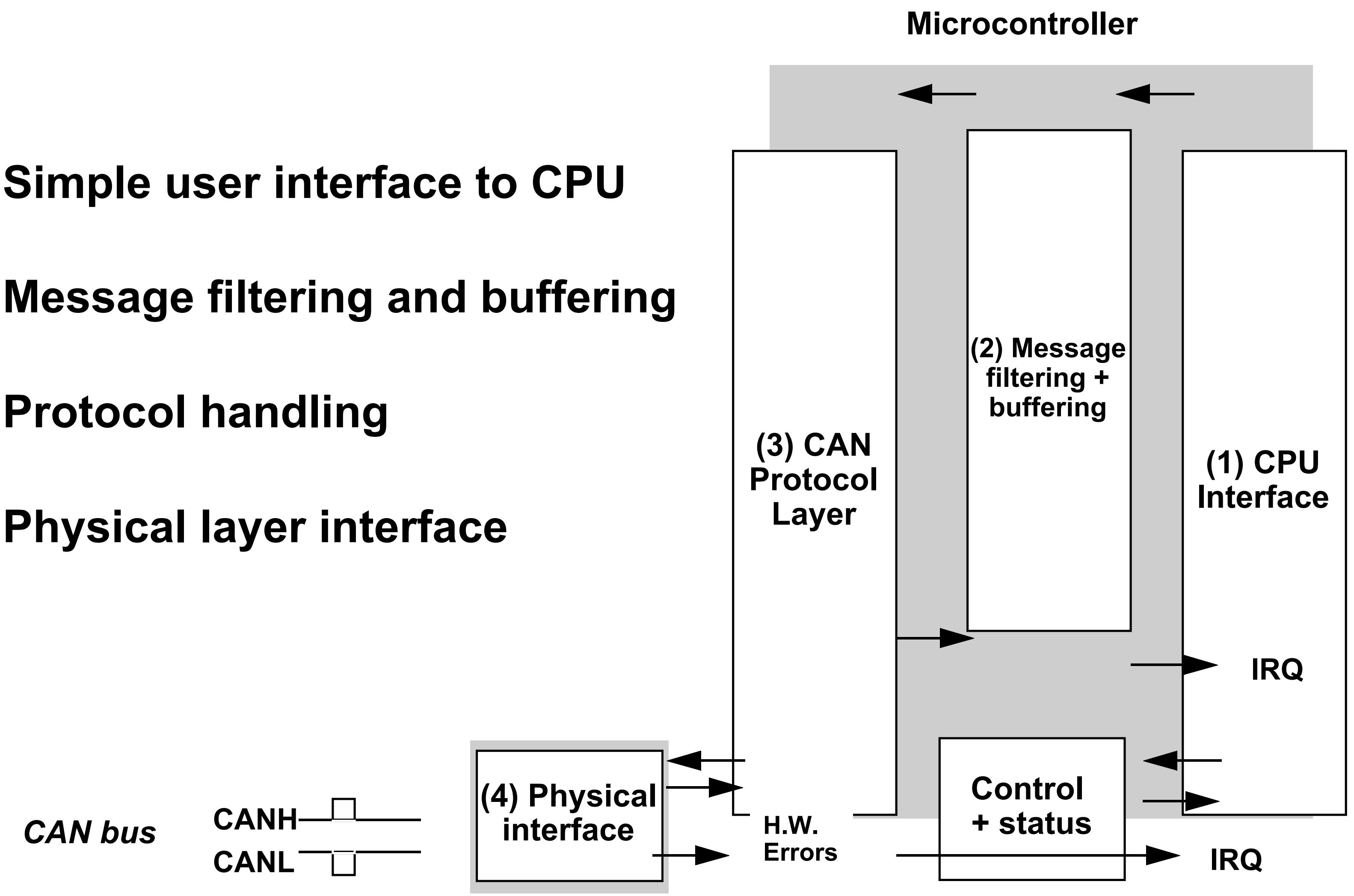
- Causes of an Overload Frame
  - A receiver node needs more time to process current data before receiving the next Message Frame .
  - The detection of a dominant bit during the INTermission field

Technical Overview

CAN Controller Requirements
FullCAN vs BasicCAN
BasicCAN Receive Structures

Requirements of a CAN Controller

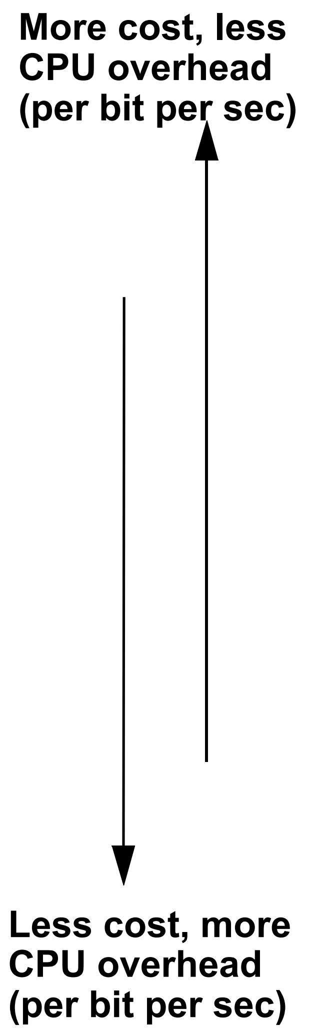
- Simple user interface to CPU
- Message filtering and buffering
- Protocol handling
- Physical layer interface



FullCAN vs BasicCAN

- **FullCAN Controller:**
  - Typically 16 message buffers, sometimes more.
  - Global and Dedicated Message Filtering Masks
  - Dedicated H/W for Reducing CPU Workload
  - More Silicon => more cost
    - » e.g. Powertrain

- **BasicCAN Controller:**
  - 1 or 2 Tx and Rx buffers
  - Minimal Filtering
  - More Software Intervention
  - Low cost
    - » e.g. Car Body


More cost, less CPU overhead (per bit per sec)
Less cost, more CPU overhead (per bit per sec)

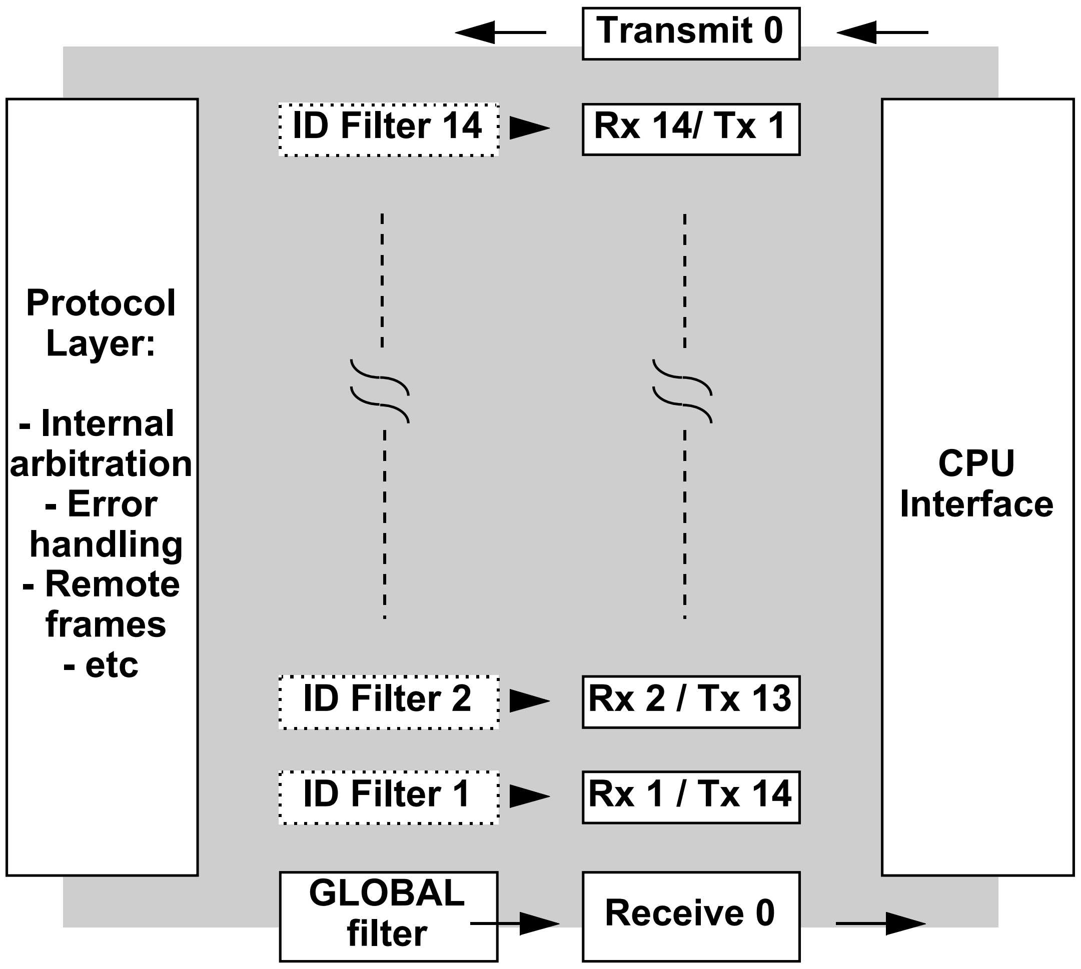
A FullCAN Controller

Advantages:

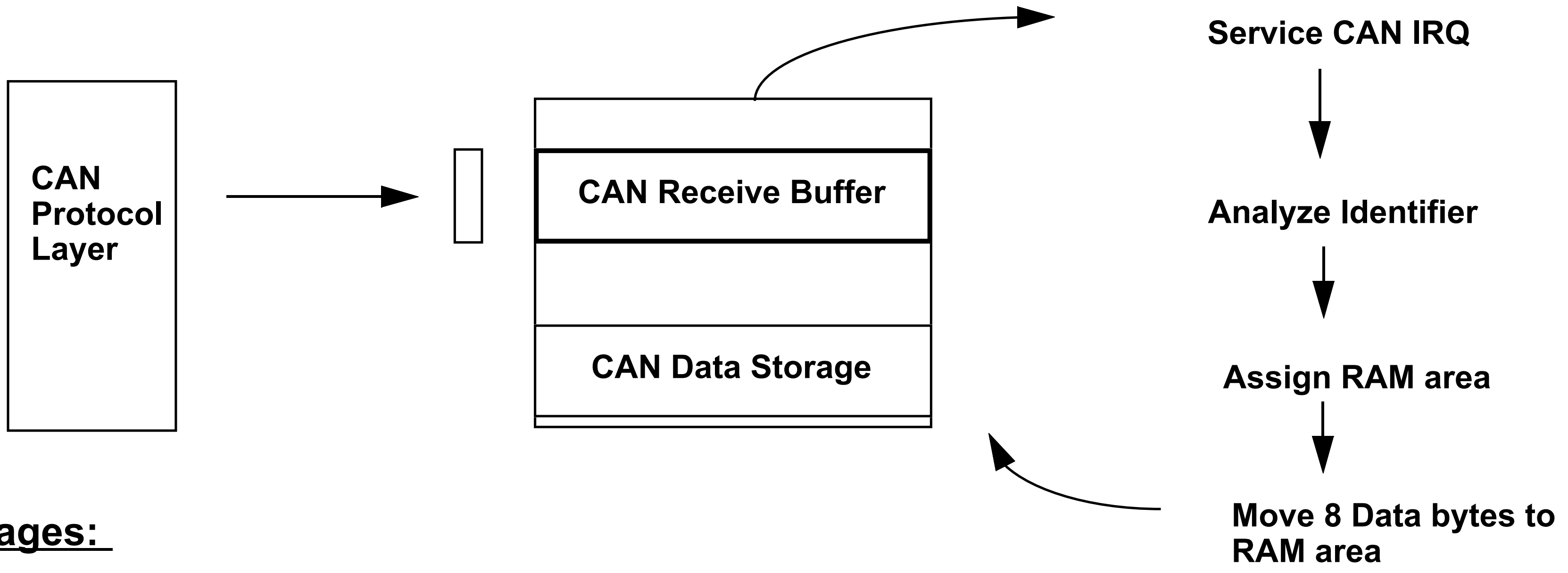
- LESS software intervention since data storage already assigned
- LESS time critical

Disadvantages:

- Loses efficiency if > 14 ID's
  - Leads to worst case interrupt loading
- More silicon required than Basic CAN => Higher cost



BasicCAN
Method 1, Rx Structure



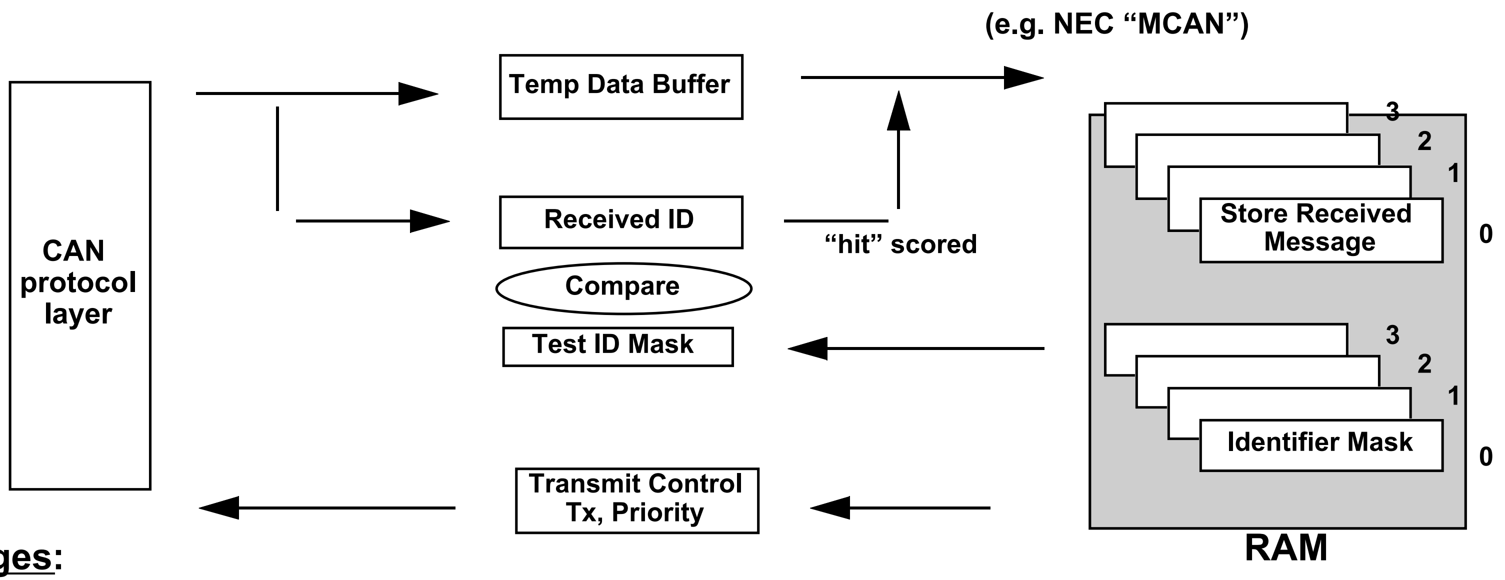
Advantages:

- Low Cost
- Not a fixed link between a receive buffer and a message identifier

Disadvantages:

- Software intensive
- More likely to get an overflow due to minimal receive buffers

BasicCAN
Method 2, Rx Structure



Advantages:

- **Area of data storage is assigned in Hardware depending on ID**
- **Allows some flexibility: Less ID masks leaves more general purpose RAM / Stack space, important for high level languages**

Disadvantages:

- **MORE Time critical, Only allows a certain amount of filtering is possible, dependent on bus speed**
- **MORE silicon area**

Summary of CAN Implementations

- Full CAN is not appropriate for Car Body for cost reasons:
  - The RAM required for message buffering and filtering is very silicon intensive and expensive
- A Full CAN receiver can have worst case Rx interrupt situations similar to Basic CAN receivers.
  - A FullCAN controller with less receive buffers than there are message identifiers experiences increased loading on the globally filtered Rx buffer
    - » This is THE SAME situation as on a Basic CAN controller.

Technical Overview

MCAN
MSCAN
TouCAN

Motorola CAN Implementations

BASIC CAN: Less cost, more CPU overhead (per bit per sec)

1) MCAN (on HC05X family)

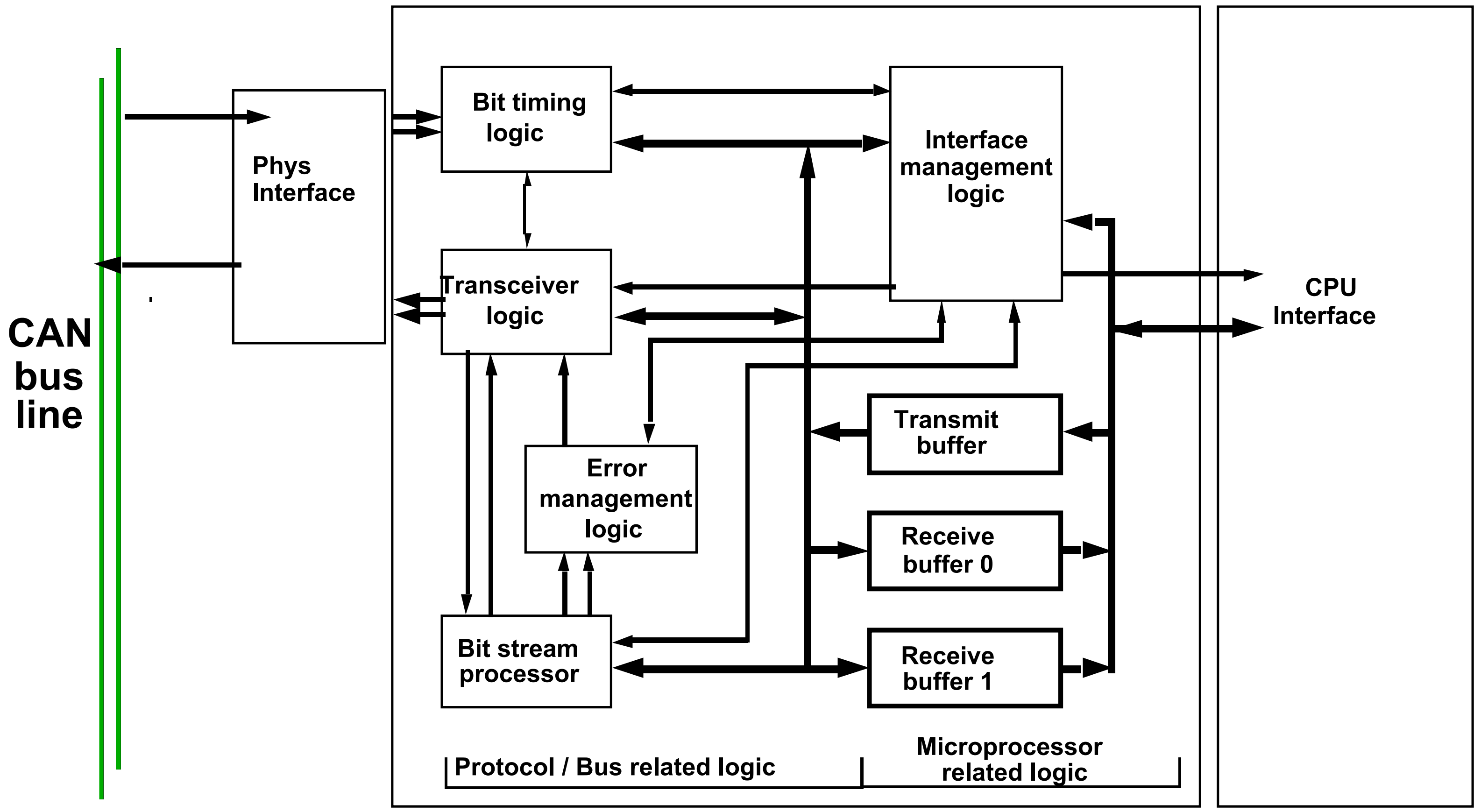
2) MSCAN08

3) MSCAN12

4) TouCAN

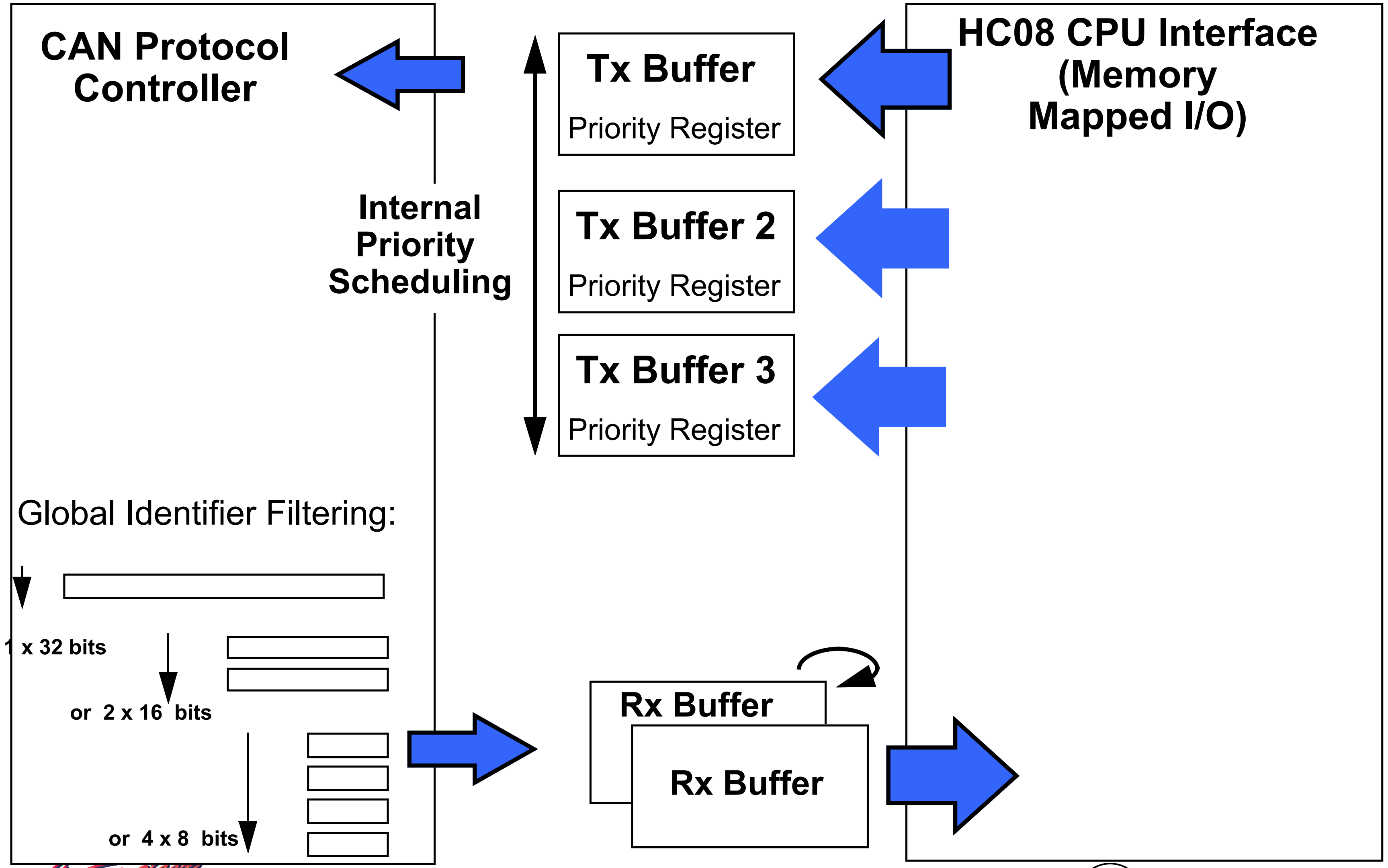
FULL CAN: More cost, less CPU overhead (per bit per sec)

MCAN HC05X Family

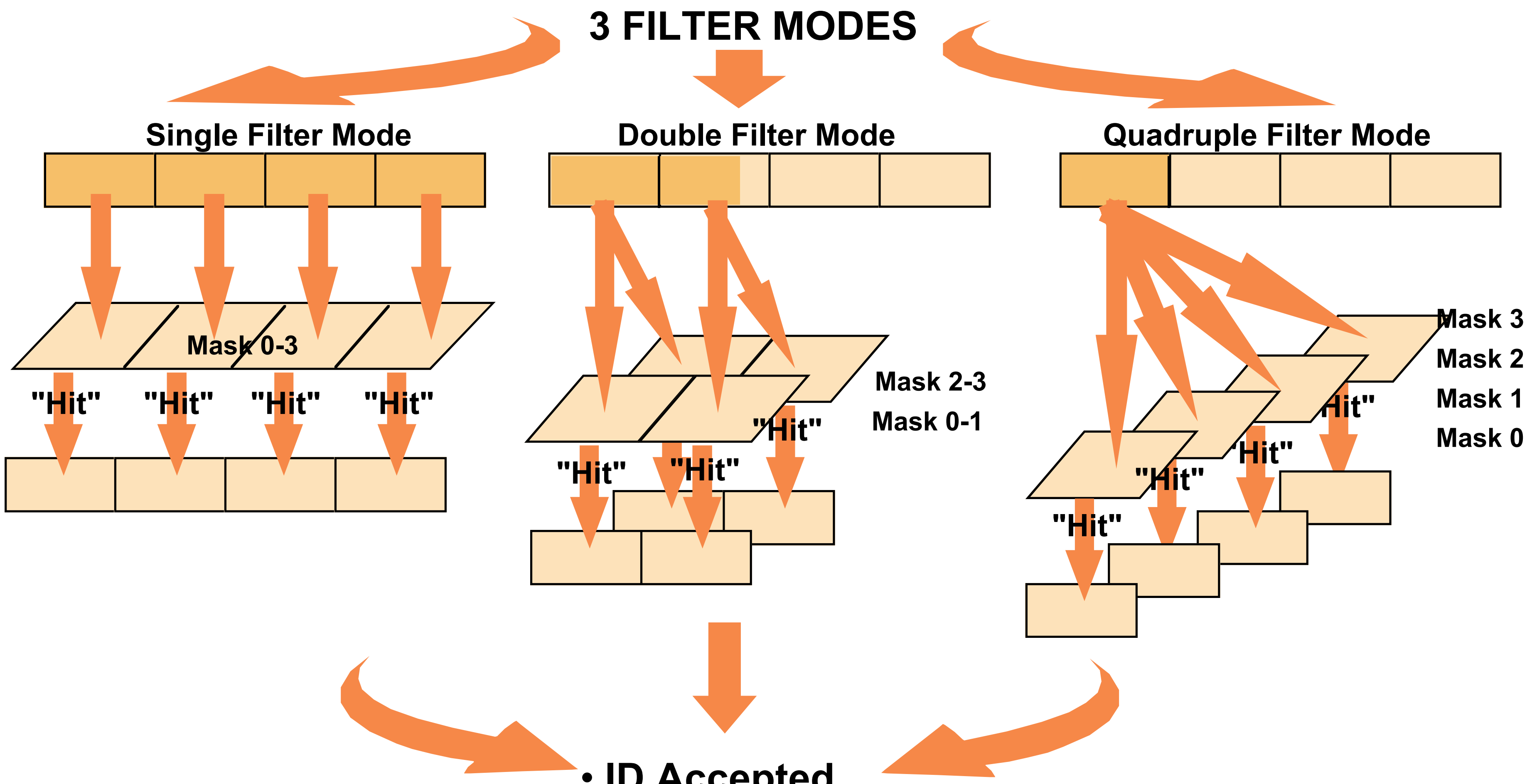


- 11 bit Identifier acceptance filter
- 2 Rx and 1 Tx buffers

MSCAN 08 Buffer and Filter Scheme



MS08CAN Filter Scheme



- ID Accepted
- CPU interrupted

Rx Interrupt CPU Load
HC05 vs HC08

HC05 - MCAN
One 11-Bit Acceptance Filter

Assumptions:

4MHz Bus Frequency
All Messages have to be accepted
80 % Bus load
average of 4 data Bytes / message
(1296 µs/message with no stuff bit)

80% Bus load --> 617 messages / s

Receive Routine: 800 cycles = 200 µs

Total Receive Interrupt Time per s:
617 * 200 µs = 123 ms

CPU load ONLY for
4MHz HC05 MSCAN
Receive: 12.3%

HC08- MSCAN
Four 8-Bit GLOBAL Acceptance Filter

Assumptions:

8MHz Bus Frequency
Only required messages accepted (70%)
80 % Bus load
average of 4 data Bytes / message
(1296 µs/message with no stuff bit)

80% Bus load, 70% accepted messages
--> 431 messages / s

Receive Routine: 650 cycles = 81.25 µs

Total Receive Interrupt Time per s:
431 * 81.25 µs = 35 ms

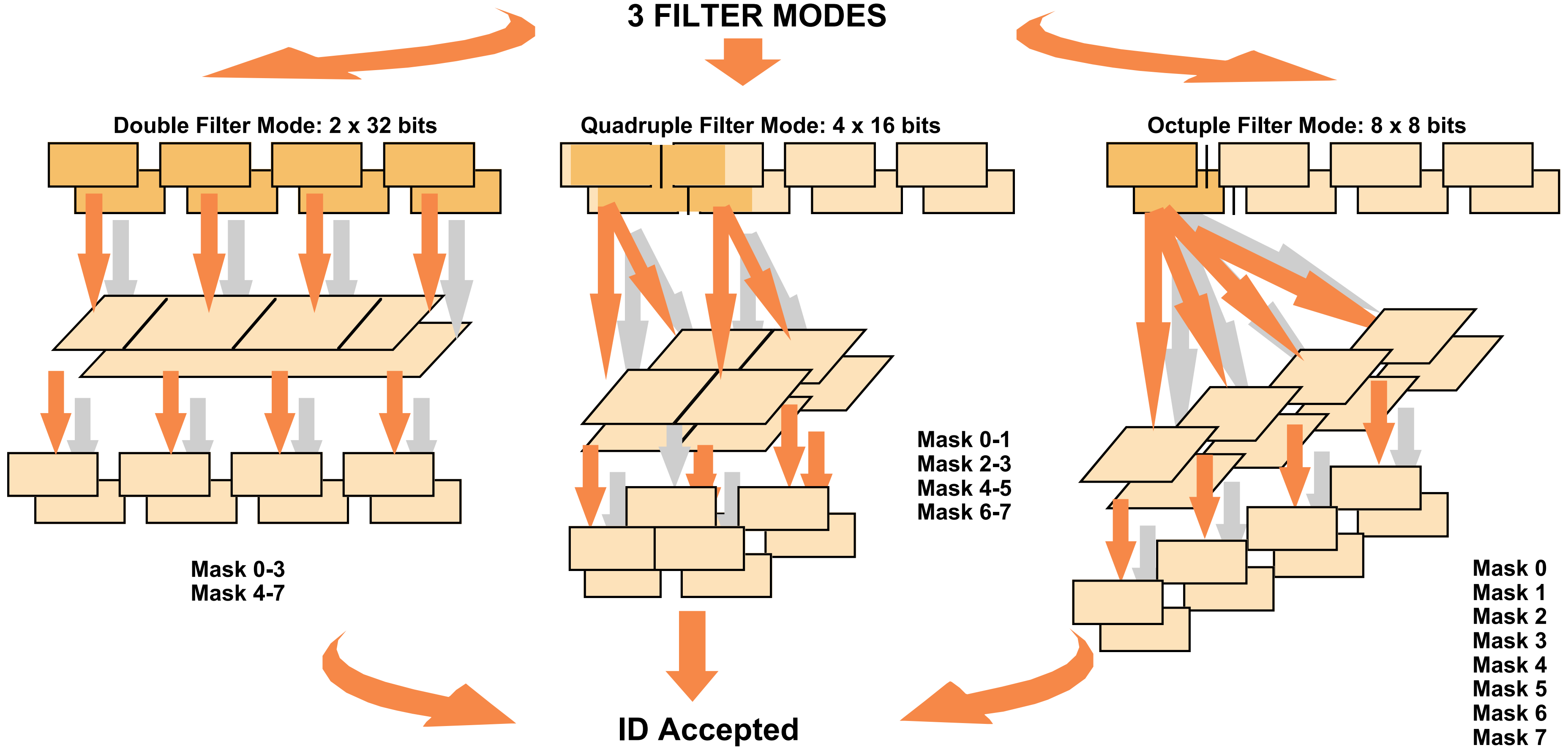
CPU load ONLY for
8MHz HC08 MSCAN
Receive: 3.5%

Team Automotive

MOTOROLA

MSCAN12 Filter Scheme

- DOUBLE the number of filters of MSCAN08
- Further reduces CPU interrupt loading
- Otherwise identical to MSCAN08



Rx Interrupt CPU Load
HC08 vs HC12

HC08- MSCAN08
Four 8-Bit GLOBAL Acceptance Filter

Assumptions:

8MHz Bus Frequency
Only required messages accepted (70%)
80 % Bus load
average of 4 data Bytes / message
(1296 μs/message with no stuff bit)

80% Bus load, 70% accepted messages
--> 431 messages / s

Receive Routine: 650 cycles = 81.25 μs

Total Receive Interrupt Time per s:
431 * 81.25 μs = 35 ms

CPU load ONLY for 8MHz HC08 MSCAN Receive: <u>3.5%</u>

HC12- MSCAN12
Eight 8-Bit GLOBAL Acceptance Filter

Assumptions:

8MHz Bus Frequency
Only required messages accepted (50%)
80 % Bus load
average of 4 data Bytes / message
(1296 μs/message with no stuff bit)

80% Bus load, 50% accepted messages
--> 308 messages / s

Receive Routine: 300 cycles = 37.5 μs

Total Receive Interrupt Time per s:
308 * 37.5 μs = 11.5 ms

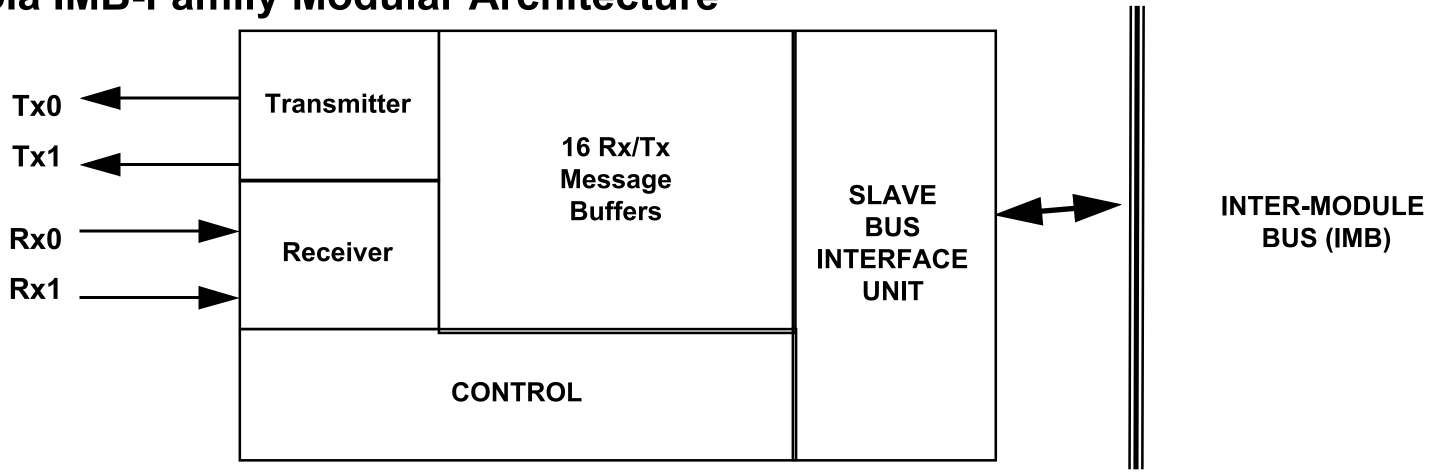
CPU load ONLY for 8MHz HC12 MSCAN12 **Receive: <u>1.15%</u>**

MSCAN Module Features

- Implementation of the CAN protocol - Version CAN 2.0A/B
  - Standard and extended data frames
  - 0 - 8 bytes data length
  - programmable bit rate up to 1MBit/s
- Double buffered receive storage system
- Triple buffered transmit storage scheme with internal prioritization using “local priority” concept
- Flexible maskable identifier filters
- Programmable wake-up functionality with integrated low-pass filter
- Programmable loop-back mode supports selftest
- Separate signaling and interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source (PLL or oscillator)
- Programmable link to on-chip timer module for time stamping or network synchronization
- Low power sleep mode

The High Performance TouCAN Module

- Full implementation of the CAN protocol - Version 2.0B
- 16 Rx/Tx Message Buffers of up to 8 bytes Data Length
- Programmable Bit Rate up to 1 Mbit/sec
- Programmable Global Receive identifier mask
- 2 Dedicated Receive identifier masks
- Programmable transmit-first scheme
- Time stamping to allow network timing synchronization
- Low power "sleep" mode, with programmable "wake up"
- Motorola IMB-Family Modular Architecture


Tx0
Tx1
Rx0
Rx1
Transmitter
Receiver
16 Rx/Tx Message Buffers
SLAVE BUS INTERFACE UNIT
CONTROL
INTER-MODULE BUS (IMB)