“Migrating from DCAN to MCAN”

Sahil Deshpande (Platform Software)

Hareesh Janakiraman (Applications Engineering- C2000 Microcontrollers)

Need for migration

- C2000 devices feature three types of Controller Area Network (CAN) modules: eCAN, DCAN, and MCAN.
- None of the three CAN modules are software compatible with each other.
- Some C2000 devices only feature MCAN as it supports both Classic CAN and CAN-FD.
- DCAN and MCAN modules employ a completely different architecture and hence, a *very* different programming approach is warranted between the modules.
- The most prominent differences between DCAN and MCAN have been highlighted in this video.

Basic differences between DCAN and MCAN

Feature	DCAN	MCAN
Bit-rate	Fixed bit-rate for the entire frame	Two bit-rates can be used: a slower bit-rate for the *nominal phase* and a faster bit-rate for the *data phase*
Transmission speed	Capped at 1Mbps	Up to 1Mbps can be used for the *nominal phase* and up to 5Mbps for the *data phase*
Number of bytes transmitted per frame (Payload capability)	Any number of bytes from 0 to 8 can be transmitted	In addition to 0 to 8 bytes, transmission of 12/16/20/24/32/48/64 data bytes is possible
Nomenclature of data storage elements	Data is stored in Message Objects. Message objects are also sometimes referred to as Mailboxes.	Data is stored in buffers tied to *filter elements*
Number of data storage elements	Fixed at 32, regardless of the number of bytes to be transmitted or received	Depending on the configuration of the element, the number of buffers is flexible
CRC-field length	15 bits	15, 17 or 21-bit CRC
Time-stamping support	No	Yes
Transmitter delay compensation	Not required	Required for faster bit-rates in the data phase

Bit-timing configuration

- DCAN employs a single bit-rate for the entire frame.
- MCAN employs a slower “nominal” bit-rate and a faster “data” bit-rate. This warrants programming two registers, MCAN_NBTP and MCAN_DBTP, during module initialization.
- The faster bit-rate for the data phase also warrants *Transmitter Delay Compensation* (TDC) without which bit errors can occur.

DCAN Message RAM

- DCAN Message RAM has a fixed configuration and contains up to 32 message objects.

- Each message object has the same structure, and, can be configured as either a transmit message object or a receive message object.

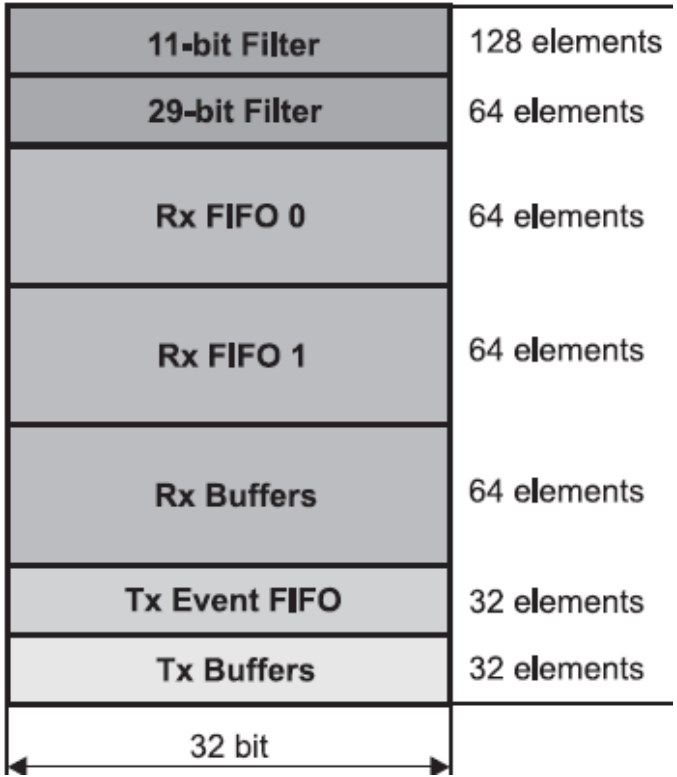
Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

MCAN Message RAM

Message RAM can be configured to have up to:

- 128 standard filter elements
- 64 extended filter elements
- 64 Rx FIFO 0 and 64 Rx FIFO 1 elements
- 64 Rx buffer elements
- 32 Tx event FIFO elements
- 32 Tx buffer elements

(where each of the above have distinct structures)



Configuring Message RAM

- The design of MCAN Message RAM offers tremendous flexibility, enabling allocation of the available memory to each of the sections based on the application needs.
- Message RAM Configuration involves configuring the following:
  - Starting Address of each section (32-bit addresses)
  - Size (number of elements) of each section
  - Size of elements in each section (only for Rx FIFO and Rx/Tx Buffer Elements)

Message RAM configuration - Precautions

- The MCAN module does not check for overlap between sections. You have to ensure that the sections are stored in distinct regions to prevent memory corruption.
- The MCAN module addresses the Message RAM in 32-bit words. Therefore, the length of each section is a multiple of 32-bits.
- Since Message RAM is configurable according to the application needs, RAM is used more optimally. This advantage comes with the requirement for correct configuration.

C2000ware macros solution

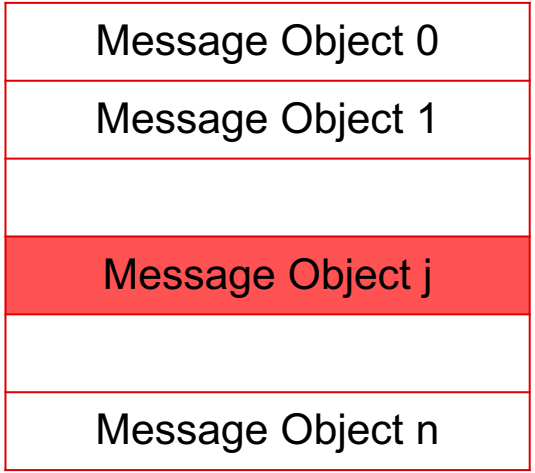
- C2000ware examples offer a defined set of macros, which based on user configurations, calculates a permissible set of starting addresses for each sections that can be used for the Message RAM Configuration.

```
//
// Defines
//
#define MCAN_STD_ID_FILTER_NUM          (1U)
#define MCAN_EXT_ID_FILTER_NUM          (0U)
#define MCAN_FIFO_0_NUM                 (5U)
#define MCAN_FIFO_0_ELEM_SIZE           (MCAN_ELEM_SIZE_64BYTES)
#define MCAN_FIFO_1_NUM                 (10U)
#define MCAN_FIFO_1_ELEM_SIZE           (MCAN_ELEM_SIZE_64BYTES)
#define MCAN_RX_BUFF_NUM                (10U)
#define MCAN_RX_BUFF_ELEM_SIZE          (MCAN_ELEM_SIZE_64BYTES)
#define MCAN_TX_BUFF_SIZE               (10U)
#define MCAN_TX_FQ_SIZE                 (0U)
#define MCAN_TX_BUFF_ELEM_SIZE          (MCAN_ELEM_SIZE_64BYTES)
#define MCAN_TX_EVENT_SIZE              (10U)

//
//  Defining Starting Addresses for Message RAM Sections,
//  (Calculated from Macros based on User defined configuration above)
//
#define MCAN_STD_ID_FILT_START_ADDR     (0x0U)
#define MCAN_EXT_ID_FILT_START_ADDR     (MCAN_STD_ID_FILT_START_ADDR + ((MCAN_STD_ID_FILTER_NUM * MCANSS_STD_ID_FILTER_SIZE_WORDS * 4U)))
#define MCAN_FIFO_0_START_ADDR          (MCAN_EXT_ID_FILT_START_ADDR + ((MCAN_EXT_ID_FILTER_NUM * MCANSS_EXT_ID_FILTER_SIZE_WORDS * 4U)))
#define MCAN_FIFO_1_START_ADDR          (MCAN_FIFO_0_START_ADDR + (MCAN_getMsgObjSize(MCAN_FIFO_0_ELEM_SIZE) * 4U * MCAN_FIFO_0_NUM))
#define MCAN_RX_BUFF_START_ADDR         (MCAN_FIFO_1_START_ADDR + (MCAN_getMsgObjSize(MCAN_FIFO_1_ELEM_SIZE) * 4U * MCAN_FIFO_1_NUM))
#define MCAN_TX_BUFF_START_ADDR         (MCAN_RX_BUFF_START_ADDR + (MCAN_getMsgObjSize(MCAN_RX_BUFF_ELEM_SIZE) * 4U * MCAN_RX_BUFF_NUM))
#define MCAN_TX_EVENT_START_ADDR        (MCAN_TX_BUFF_START_ADDR + (MCAN_getMsgObjSize(MCAN_TX_BUFF_ELEM_SIZE) * 4U * (MCAN_TX_BUFF_SIZE + MCAN_TX_FQ_SIZE)))
```

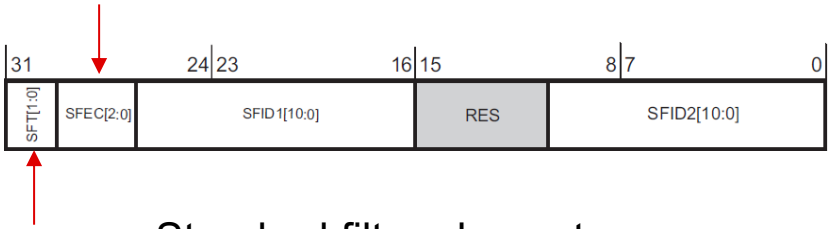
Filtering in DCAN

Received Standard ID CAN Frame

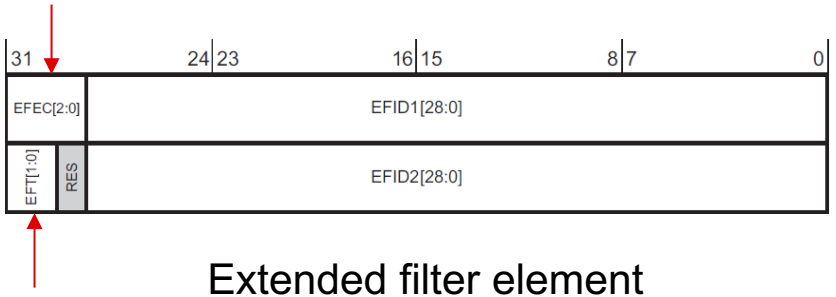


Filter elements in MCAN

- Filter elements are defined structures, which need to be configured in the Message RAM.
- Filter element behavior is determined by the following:
  - Filter type
  - Filter element configuration



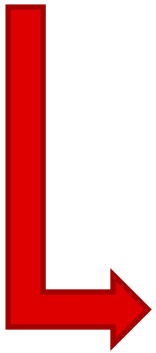
Standard filter element



Extended filter element

Filtering in MCAN

Received Standard ID CAN Frame



Standard Filter List

Standard Filter Element 0
Standard Filter Element 1
Standard Filter Element j
Standard Filter Element n

Reject Frame
Store in Rx Buffer Element p
Store in Rx FIFO 0
Store in Rx FIFO 1

Configuring filter elements

- During module initialization, filter elements can be added using *Driverlib* APIs, which are stored as defined structures in the Message RAM.
- Filter types:
  - Range filter
  - Dual ID filter
  - Classic bit mask filter
- Note: Only exact matching IDs can be stored in Rx buffers, matching frames for all other filter types need to be stored in Rx FIFOs.

DCAN FIFOs

- In DCAN, multiple receive message objects can be concatenated to a FIFO buffer.
- The process of reading from a FIFO is exactly similar to reading from a receive message object.
- In order to achieve true FIFO functionality, all message objects of the FIFO need to be read and cleared, due to the implicit priority of the message objects.

MCAN FIFOs

- MCAN offers a number of FIFOs that are separate, defined structures stored in the Message RAM, that offer more flexibility and ease of use in applications.
- The structures available in MCAN are as follows:
  - 2 x Rx FIFOs (Rx FIFO 0 & Rx FIFO 1)
  - Tx FIFO
  - Tx Event FIFO
- Each FIFO has a corresponding register to store status information including Fill Level, Put Index and Get Index to aid in functioning.

Key difference in FIFO handling

	DCAN FIFO	MCAN FIFO
Storing received message	Lowest available message object	Put Index
Reading oldest message	Depends on application	Get Index

CAN Resources

- CAN Training Module C2000 Academy
  (https://dev.ti.com/tirex/global?id=c2000Academy)
- TI Precision Labs CAN overview *(https://training.ti.com/ti-precision-labs-canlinsbc-can-and-can-fd-overview)*
- CAN Application Reports
  - Getting Started with the MCAN (CAN FD) Module
    (http://www.ti.com/lit/SPRACU9)
  - DCAN to MCAN Migration Guide
    (http://www.ti.com/lit/SPRAD59)