# CAN Controller and Transceiver with SPI / I²C Interface

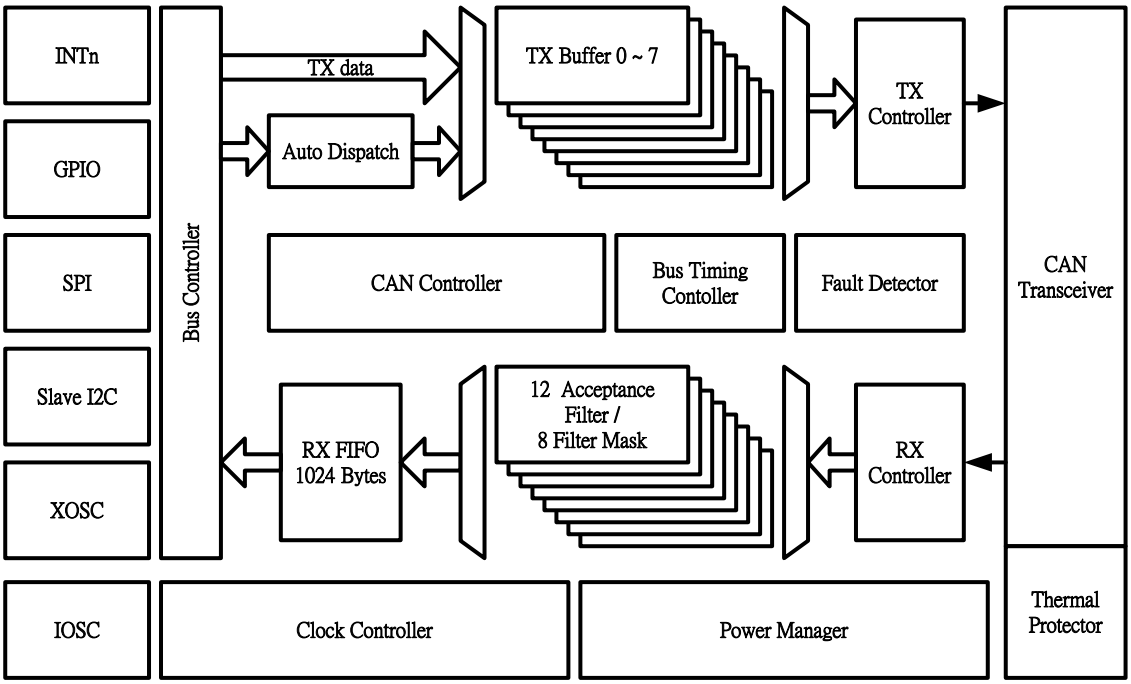
## GENERAL DESCRIPTION

The IS31IO8972 is a stand-alone Controller Area Network (CAN) protocol controller with the embedded CAN transceiver. It supports CAN 2.0B standard and the maximal bit rate is 1 Mb/s. It is capable of transmitting and receiving standard and extended message frames. It includes eight independent transmit buffers with auto-dispatch function and 1024-byte receive FIFO with 12 ID acceptance filtering and message management. The MCU communication is implemented via an industry standard Serial Peripheral Interface (SPI) and I²C bus.

## FEATURES

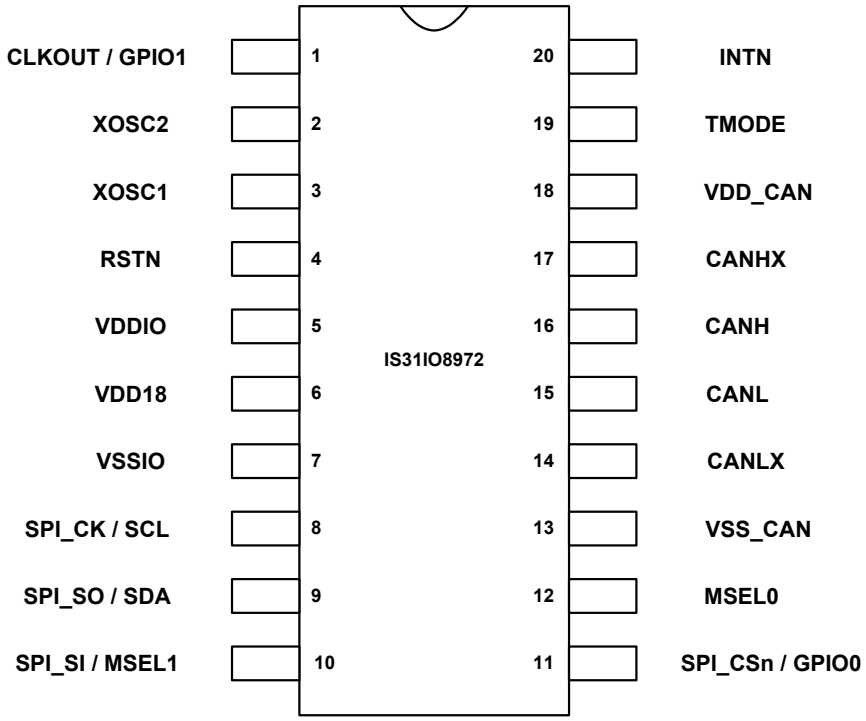
- Single + 5V power supply
- Maximal operating clock up to 24MHz
- Built-in one CAN controller according to CAN protocol version 2.0B.
  - 0~8 byte message length
  - Standard and extended data frames
  - Programmable bit rate up to 1Mb/s
  - Support for remote frames
  - 1024 Bytes receive FIFO
  - 12 full acceptance filters
  - 8 full filter masks
  - 8 transmit buffers with abort feature
  - Auto-dispatch function for each transmit buffer
  - Listen mode
  - Loop-back mode for self-test operation
- Built-in CAN transceiver full support CAN v2.0B specification
  - Built-in IOSC up to 16MHz
  - Hardware interface
  - High speed SPI interface up to 3Mb/s bit rate
  - Supports SPI mode 0,0 and 1,1
  - High speed Slave I²C interface up to 2Mb/s bit rate
  - Interrupt output pin with selectable enables
- Low power CMOS technology
  - 50mA active current typical
  - 1mA standby current
- External wake-up by SPI/ I²C and CAN receiver
- Industrial operating temperature range (-40°C ~ +125°C)
- 20-pin SSOP packages
- RoHS compliance

## BLOCK DIAGRAM




**Block Diagram of IS31IO8972**

## PACKAGE TYPE




**IS31IO8972 20-pin SSOP**

## PIN CONFIGURATION

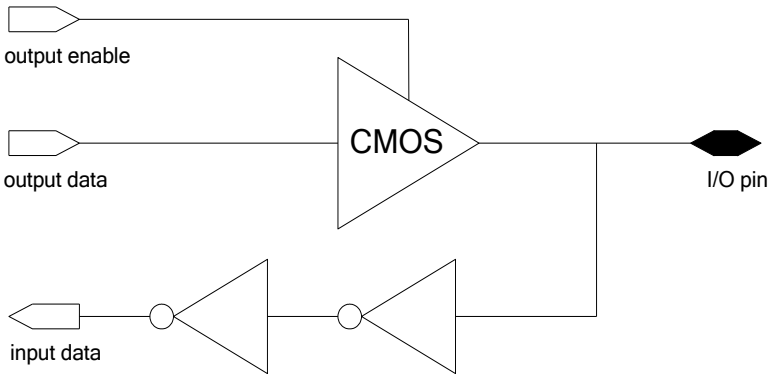
A “CMOS pin” can be used as Input or Output mode. To use these pins as output mode, S/W or H/W needs to set the corresponding output enable control bit to 1. Otherwise, the output enable control bit should clear to 0. In output mode, these pins can sink and drive at least 4mA current.

The pins functionality is described in the following table. There are no three states outputs pins and internal signals.

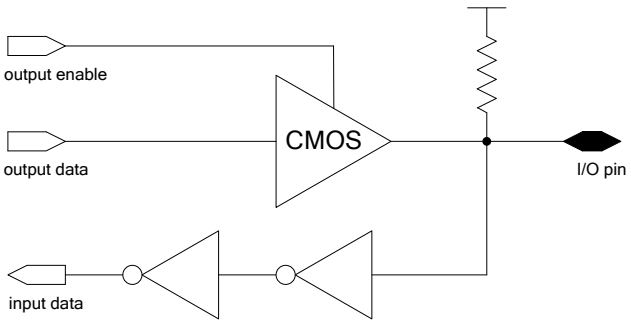
| Name | Pin No. | Default | Pin Config. | Function Description |
|---|---|---|---|---|
| CLKOUT / GPIO1 | 1 | O | B | Oscillator frequency clock output / General purpose I/O<br>The external pull-down or pull-up will determine I²C ID address during reset |
| XOSC2 | 2 | O | A | Oscillator output |
| XOSC1 | 3 | I | A | Oscillator input |
| RSTN | 4 | I | IU | Active low reset input |
| VDDIO | 5 | - | A | +3.0V ~ +5.5V chip power supply. A good decoupling capacitor between VDDIO and VSSIO pins is critical for good performance. |
| VDD18 | 6 | - | A | +1.8V positive core power output. A good decoupling capacitor between VDD18 and VSSIO pins is critical for good performance. |
| VSSIO | 7 | - | A | Ground for digital core and I/O |
| SPI_CK / SCL | 8 | I | I<br>IU | If MSEL0 is one, enable SPI mode, this pin is SPI serial clock input.<br>If MSEL0 is zero and SPI_SI/MSEL1 is zero, enable I²C mode, this pin is I²C clock input with internal pull-high |
| SPI_SO / SDA | 9 | B | O<br>BU | If MSEL0 is one, enable SPI mode, this pin is SPI serial data output.<br>If MSEL0 is zero and SPI_SI/MSEL1 is zero, enable I²C mode, this pin is I²C data bus line with internal pull-high |
| SPI_SI / MSEL1 | 10 | I | I | If MSEL0 is one, enable SPI mode, this pin is SPI serial data input<br>If MSEL0 is zero and this pin is zero, enable I²C interface<br>If MSEL0 is zero and this pin is one, enable Transceiver only mode |
| SPI_CSn / GPIO0 | 11 | I | IU<br>B | If MSEL0 is one, enable SPI mode, this pin is SPI chip select (active low) with internal pull-high.<br>If MSEL is zero, this pin is a general purpose I/O<br>External pull-down or pull-up will determine I²C ID address during reset |
| MSEL0 | 12 | I | I | Host mode select input,<br>1:SPI mode, 0: I²C or Transceiver only mode |
| VSS_CAN | 13 | - | A | Ground for CAN transceiver |
| CANLX | 14 | - | A | Common-mode stabilization output of CANL |
| CANL | 15 | - | A | Low-level CAN bus line |
| CANH | 16 | - | A | High-level CAN bus line |
| CANHX | 17 | - | A | Common-mode stabilization output of CANH |
| VDD_CAN | 18 | | A | +5V positive power supply for CAN transceiver. A good decoupling capacitor between VDD_CAN and VSS_CAN pins is critical for good performance. |
| TMODE | 19 | I | I | Test mode enable, 1:ATPG mode, 0:Normal mode |
| INTn | 20 | O | O | Interrupt output (active low) |

**Table: PAD Description**

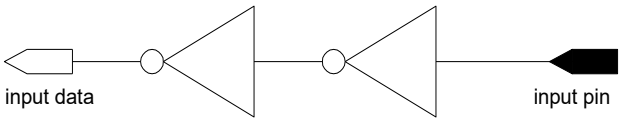
**Note: Pin Configuration description**



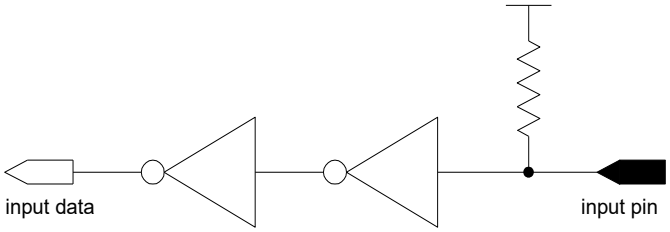

**I/O Configure Type 'B': Standard CMOS bi-directional pad.**



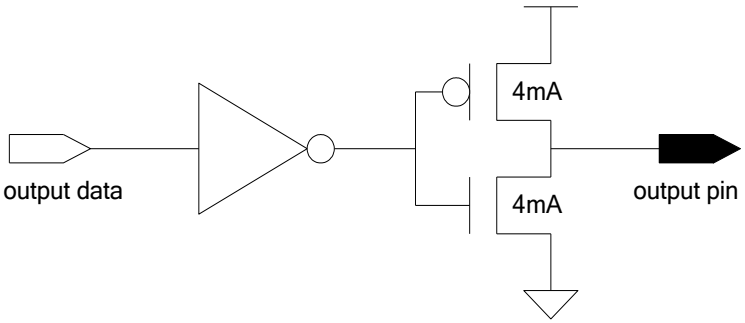

**I/O Configure Type 'BU': Standard CMOS bi-directional pad with pull-high resistor.**



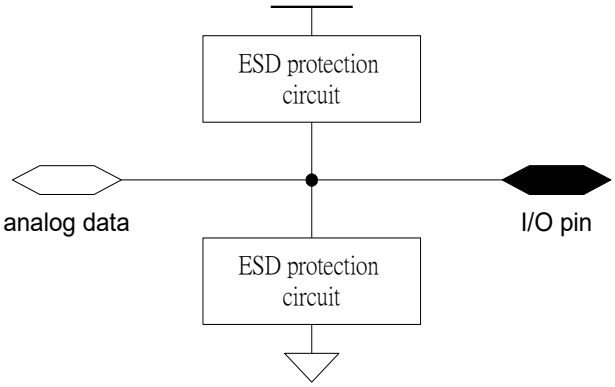

**I/O Configure Type 'I': Input only pad.**




**I/O Configure Type 'IU': Input only pad with an internal pull-up resistor.**




**I/O Configure Type 'O': Output only pad.**

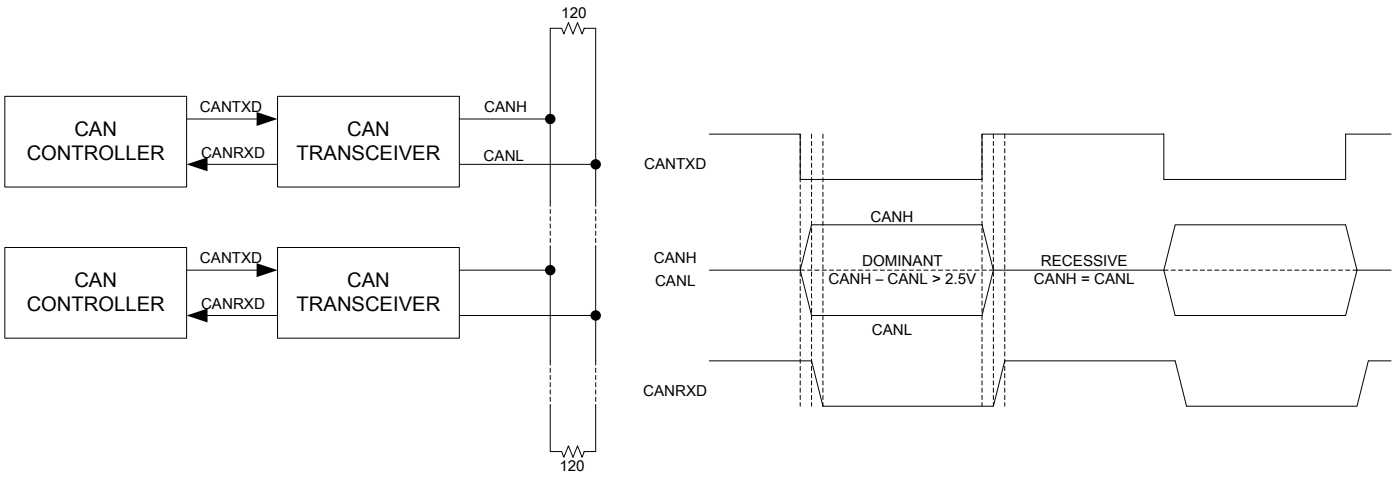



**I/O Configure Type 'A': Analog pad**

# 1. CAN CONTROLLER

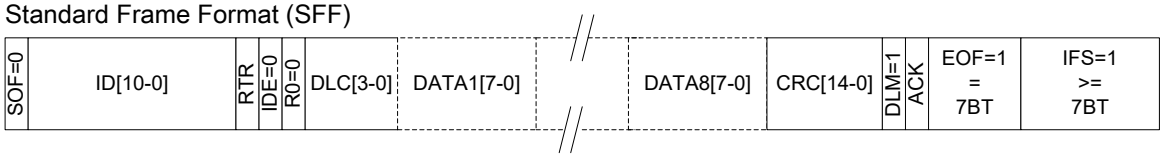
## 1.1 Brief Introduction to CAN

The Controller Area Network (CAN) is a serial asynchronous bus allowing multi-master communications. The CAN bus uses a single terminated twisted pair with maximum data rate of 1MBps. Typical length of the bus can be up to 40 meters. One unusual aspect of the CAN protocol is its message based nature. Nodes on the bus do not have specific address instead the messages have unique identifiers which are used for determining its priority. Each node depending on its functionality transmits specific types of messages and also responds to specific related messages. CAN is especially popular in connecting electronic control modules, sensors, actuators in automotive and industrial applications because its simple bus structure and fault-tolerant capability from extensive error checking.
The physical media of typical CAN bus is a twisted pair terminated with 120Ohm on both ends. The twisted pair has two signals, CANH and CANL. CANH and CANL forms a 5V pseudo differential signal thus have reliable transmissions even under high noise environment. There are two possible logic states on the bus. Dominant state (logic 0) is when CANH is actively pulled to 5V and CANL is actively pulled to 0V. Recessive state (logic 1) is when the bus is not actively driven and both CANH and CANL are pulled toward 2.5V by the termination resistors. The bus also exhibits the wire-AND characteristics, that is bus is in recessive state if only if all nodes are recessive, and bus is in dominant state if one or more nodes are dominant. Because of this pseudo differential nature, it is possible to maintain correct data transmission even if one of the wires is broken. To drive the CAN bus, typically an external transceiver is used to provide isolated drive.

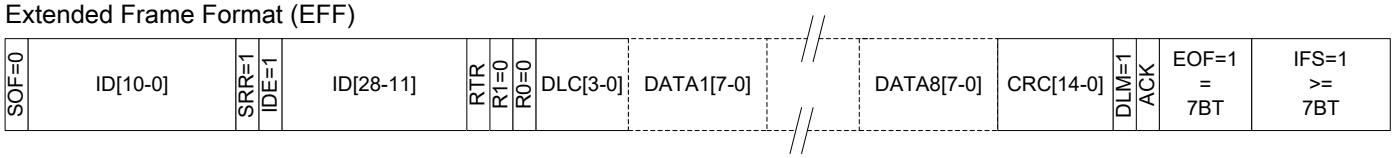



The CAN protocol is a modified version of Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) similar to Ethernet. If two nodes are trying to send at the same time, instead of collision avoidance, the ID field of message will resolve itself which allows the higher priority message to continue and lower priority message been postponed. The typical standard CAN frame format and extended format are shown in the following

Standard Frame Format (SFF)




Extended Frame Format (EFF)



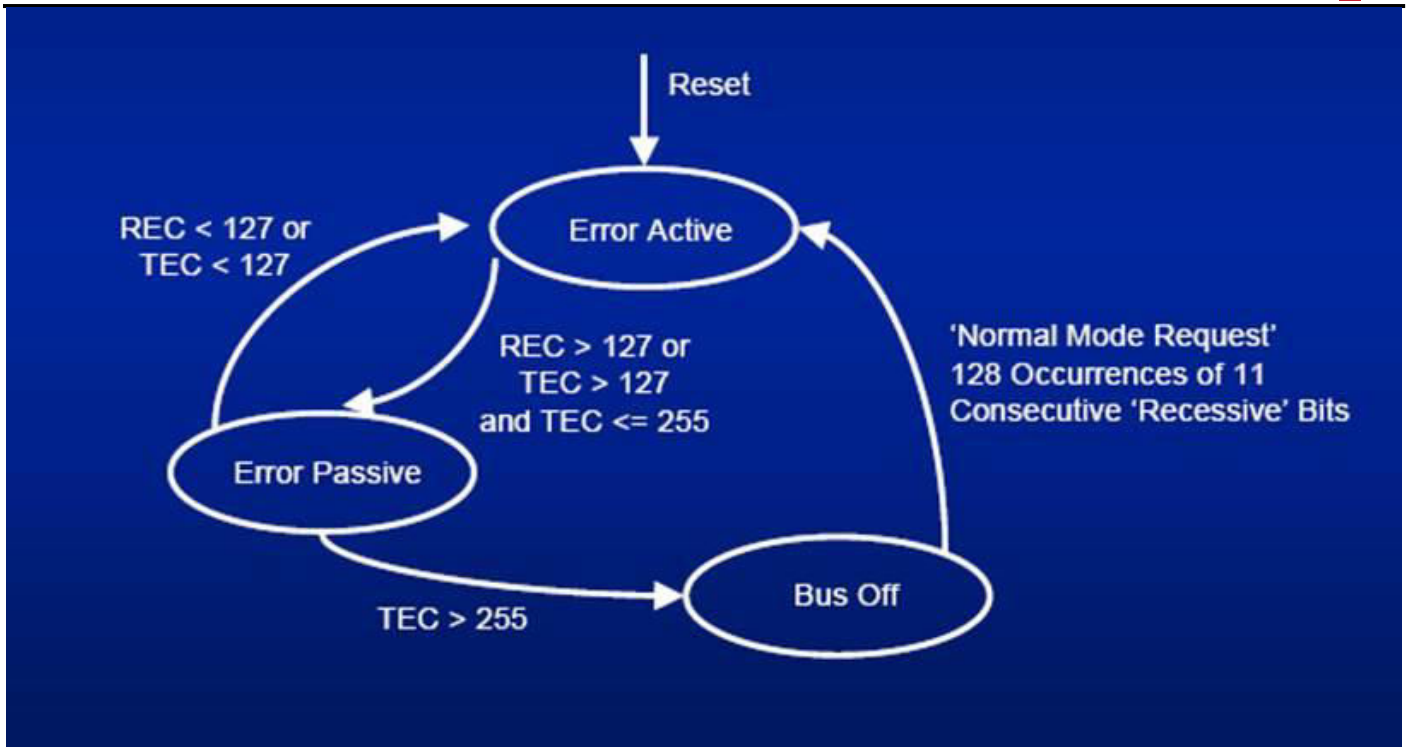

The frame starts with a bit 0 of Start of Frame (SOF), and followed by a 11-bit ID field. Because the dominant state transmitted by one node will overwrite any other nodes transmitting recessive state, the message ID with lowest number will win the bus arbitration. After arbitration, RTR bit indicate whether this is a transmit message (RTR=0), or a Remote Transmission Request (RTR=1) message. R1 and R0 are two reserve bits which should both be 0. The Data Length Code (DLC) indicates the byte count of following data. DLC must be less than 9. After data field, a 15-bit CRC is appended, and a CRC delimiter is the last bit of CRC field. The receive node will acknowledge the successful reception by sending 0 during the ACK slot. An ACK delimiter and along with minimum 7 bit time of EOF field is ensured for the bus to enter into bus IDLE state. And a minimum 3 BT (typically implemented as 7 BT) of recessive state and bus IDLE state should be used for inter-frame spacing.

The message format can be further categorized into standard (SFF) or extended format (EFF). The difference in SFF and EFF is in IDE bit. If IDE=dominant, the message is SFF, and if IDE=recessive, the message is EFF. The extended format provides 29-bit ID addressing whit the additional 18 bit ID following the IDE bit. In terms of purpose of message, the message can be considered as either data frame or remote frame. A data frame carries data sent by the transmitter to receivers (plural here). A remote frame is transmitted by a node to request other nodes to respond with data frame using same message ID. Both data frame and remote frame can take either standard or extended format.

CAN bus uses NRZ coding. To ensure enough edges for synchronization, Bit-Stuffing is used. An opposite polarity bit is inserted after 5 consecutive bits of the same level. Bit-Stuffing starts with SOF and ends with CRC (before CRC delimiter). The initial falling edge of SOF provides a hard synchronization on the start of the message.

CAN bus include many error checking mechanisms to ensure reliable operations. This includes CRC checking built-in in the message frame. CRC is calculated over the non-stuffed bit stream data starting with SOF and ends with last bit of data field. In case of CRC mismatch, the frame is discarded, and ACK bit is not responded. Nodes on the bus will also detect frame format error, bit stuffing errors. All these error been detected, a node will transmit an error frame. The error frame consists of 6 consecutive dominant bits as error flag and may be extended to 12 bits when multiple nodes are responding to the same error. This violation of bit-stuffing (more than 5 consecutive same bits) will ensure all nodes on the bus will detect the error and the error flag thus might be further extended up 12 bit times by those nodes detecting the same error. After error flag, the error frame is terminated with minimum 3 bit time of recessive state as error delimiter. Another special frame with the same format as error frame is the overload frame which is sent by a receiving node in response to a remote request message when this node can not meet the response requirement. Overload frame is also sent by nodes when detecting a violation of frame intermission rule (each frame must be separated by 3 bit time of recessive bits).

There are three fundamental states for error handling in each CAN node. This is shown in the following state diagram. REC is receive error count, and TEC is transmit error count.




The node is in normal operation when in Error Active state. In this state, the node can send all frames including active error flag with six consecutive 'dominant' bits. When error count reaches 127, it enters Error Passive states, and the node can send all frames and passive error flag with six consecutive 'recessive' bits. When error count exceeds 255, the node enters Bus Off state and is isolated from the bus. This mechanism ensures a faulty node does cause bus dead-lock. The exit of Bus Off state can only be done by re-initialization or if auto recovery is enabled when 128 occurrence of 11 consecutive recessive bit.

## 1.2 Features of CAN Controller

The CAN controller is compatible with CAN 2.0A and 2.0B standards. The controller can be configured as normal operating mode or listen-only mode. A self-test loop-back mode can also be enabled to perform internal loop-back test of the CAN controller. The receiver includes four acceptance filters that are used for ID filtering. The matched messages are stored in receive FIFO shared with CPU XRAM. The transmitter includes a 13-byte transmit buffer in XFR consisting of frame information, 4 bytes of message ID, and 8 bytes of message data. The transmitter also includes three dispatchers. The dispatcher is used for automatically transmit a pre-determined message at a fixed programmable time interval without CPU intervention. Each dispatcher also include a 13-byte transmit buffer in XFR similar to the main transmit path.
After reset, the CAN controller is put in reset mode with all state-machines forced in the initialization states. The user must exit this reset mode by setting CANRST=0 to enter into operating modes.

In IS31IO8972, the CAN controller has the main function as follows:
Operating mode
Reset mode
Listen-Only mode
Bus Auto-Recovery
Self-test mode
Self-reception
Programmable Baud-Rate
8 transmit buffers for standard and extended message frames
Transmit dispatch
8 acceptance filter and mask for receiving frame
Error detection
Loop test

## 1.3 Registers Map

IS31IO8972 uses 8-bit address to indicate all control, status and data SFRs of CAN controller. The undefined address reserved is for future use. Reading and writing operation to those addresses has no effect. Some registers are accessible in operating mode and some registers are accessible in Reset mode.

| Address (8-bit) | Operating Mode | | Reset Mode | |
|---|---|---|---|---|
| | read | write | read | write |
| 00H | Mode | Mode | Mode | Mode |
| 01H | 00H | Command Register | 00H | - |
| 02H | Status Register | - | Status Register | - |
| 03H | Interrupt Flag | - | 00H | - |
| 04H | Interrupt Enable | Interrupt Enable | Interrupt Enable | Interrupt Enable |
| 05H | TX Status 0 | TX Request 0 | 00H | - |
| 06H | TX Status 1 | TX Request 1 | 00H | - |
| 07H | Arbitration Lost | - | Arbitration Lost | - |
| 08H | Error Code | - | Error Code | - |
| 09H | Bus Timing 0 | - | Bus Timing 0 | Bus Timing 0 |
| 0AH | Bus Timing 1 | - | Bus Timing 1 | Bus Timing 1 |
| 0BH | Error Warming Limit | - | Error Warming Limit | Error Warming Limit |
| 0CH | RX Error Counter | - | RX Error Counter | RX Error Counter |
| 0DH | TX Error Counter | - | TX Error Counter | TX Error Counter |
| 0EH | Dispatch Timer 0 Low | - | Dispatch Timer 0 Low | Dispatch Timer 0 Low |
| 0FH | Dispatch Timer 0 High | - | Dispatch Timer 0 High | Dispatch Timer 0 High |
| 10H | RX Filter Number | - | RX Filter Number | - |
| 11H | RX Information | - | RX Information | - |
| 12H | RX ID-3 | - | RX ID-3 | - |
| 13H | RX ID-2 | - | RX ID-2 | - |
| 14H | RX ID-1 | - | RX ID-1 | - |
| 15H | RX ID-0 | - | RX ID-0 | - |
| 16H | RX DATA-0 | - | RX DATA-0 | - |
| 17H | RX DATA-1 | - | RX DATA-1 | - |
| 18H | RX DATA-2 | - | RX DATA-2 | - |
| 19H | RX DATA-3 | - | RX DATA-3 | - |
| 1AH | RX DATA-4 | - | RX DATA-4 | - |
| 1BH | RX DATA-5 | - | RX DATA-5 | - |
| 1CH | RX DATA-6 | - | RX DATA-6 | - |
| 1DH | RX DATA-7 | - | RX DATA-7 | - |
| 1EH | Dispatch Timer 1 Low | - | Dispatch Timer 1 Low | Dispatch Timer 1 Low |
| 1FH | Dispatch Timer 1 High | - | Dispatch Timer 1 High | Dispatch Timer 1 High |
| 20H | Filter0 Acceptance-3 | - | Filter0 Acceptance-3 | Filter0 Acceptance-3 |
| 21H | Filter0 Acceptance-2 | - | Filter0 Acceptance-2 | Filter0 Acceptance-2 |
| 22H | Filter0 Acceptance-1 | - | Filter0 Acceptance-1 | Filter0 Acceptance-1 |
| 23H | Filter0 Acceptance-0 | - | Filter0 Acceptance-0 | Filter0 Acceptance-0 |
| 24H | Filter1 Acceptance-3 | - | Filter1 Acceptance-3 | Filter1 Acceptance-3 |
| 25H | Filter1 Acceptance-2 | - | Filter1 Acceptance-2 | Filter1 Acceptance-2 |

| Address (8-bit) | Operating Mode | | Reset Mode | |
|---|---|---|---|---|
| | read | write | read | write |
| 26H | Filter1 Acceptance-1 | - | Filter1 Acceptance-1 | Filter1 Acceptance-1 |
| 27H | Filter1 Acceptance-0 | - | Filter1 Acceptance-0 | Filter1 Acceptance-0 |
| 28H | Filter2 Acceptance-3 | - | Filter2 Acceptance-3 | Filter2 Acceptance-3 |
| 29H | Filter2 Acceptance-2 | - | Filter2 Acceptance-2 | Filter2 Acceptance-2 |
| 2AH | Filter2 Acceptance-1 | - | Filter2 Acceptance-1 | Filter2 Acceptance-1 |
| 2BH | Filter2 Acceptance-0 | - | Filter2 Acceptance-0 | Filter2 Acceptance-0 |
| 2CH | Filter3 Acceptance-3 | - | Filter3 Acceptance-3 | Filter3 Acceptance-3 |
| 2DH | Filter3 Acceptance-2 | - | Filter3 Acceptance-2 | Filter3 Acceptance-2 |
| 2EH | Filter3 Acceptance-1 | - | Filter3 Acceptance-1 | Filter3 Acceptance-1 |
| 2FH | Filter3 Acceptance-0 | - | Filter3 Acceptance-0 | Filter3 Acceptance-0 |
| 30H | Filter4 Acceptance-3 | - | Filter4 Acceptance-3 | Filter4 Acceptance-3 |
| 31H | Filter4 Acceptance-2 | - | Filter4 Acceptance-2 | Filter4 Acceptance-2 |
| 32H | Filter4 Acceptance-1 | - | Filter4 Acceptance-1 | Filter4 Acceptance-1 |
| 33H | Filter4 Acceptance-0 | - | Filter4 Acceptance-0 | Filter4 Acceptance-0 |
| 34H | Filter5 Acceptance-3 | - | Filter5 Acceptance-3 | Filter5 Acceptance-3 |
| 35H | Filter5 Acceptance-2 | - | Filter5 Acceptance-2 | Filter5 Acceptance-2 |
| 36H | Filter5 Acceptance-1 | - | Filter5 Acceptance-1 | Filter5 Acceptance-1 |
| 37H | Filter5 Acceptance-0 | - | Filter5 Acceptance-0 | Filter5 Acceptance-0 |
| 38H | Filter6 Acceptance-3 | - | Filter6 Acceptance-3 | Filter6 Acceptance-3 |
| 39H | Filter6 Acceptance-2 | - | Filter6 Acceptance-2 | Filter6 Acceptance-2 |
| 3AH | Filter6 Acceptance-1 | - | Filter6 Acceptance-1 | Filter6 Acceptance-1 |
| 3BH | Filter6 Acceptance-0 | - | Filter6 Acceptance-0 | Filter6 Acceptance-0 |
| 3CH | Filter7 Acceptance-3 | - | Filter7 Acceptance-3 | Filter7 Acceptance-3 |
| 3DH | Filter7 Acceptance-2 | - | Filter7 Acceptance-2 | Filter7 Acceptance-2 |
| 3EH | Filter7 Acceptance-1 | - | Filter7 Acceptance-1 | Filter7 Acceptance-1 |
| 3FH | Filter7 Acceptance-0 | - | Filter7 Acceptance-0 | Filter7 Acceptance-0 |
| 40H | Filter8 Acceptance-3 | - | Filter8 Acceptance-3 | Filter8 Acceptance-3 |
| 41H | Filter8 Acceptance-2 | - | Filter8 Acceptance-2 | Filter8 Acceptance-2 |
| 42H | Filter8 Acceptance-1 | - | Filter8 Acceptance-1 | Filter8 Acceptance-1 |
| 43H | Filter8 Acceptance-0 | - | Filter8 Acceptance-0 | Filter8 Acceptance-0 |
| 44H | Filter9 Acceptance-3 | - | Filter9 Acceptance-3 | Filter9 Acceptance-3 |
| 45H | Filter9 Acceptance-2 | - | Filter9 Acceptance-2 | Filter9 Acceptance-2 |
| 46H | Filter9 Acceptance-1 | - | Filter9 Acceptance-1 | Filter9 Acceptance-1 |
| 47H | Filter9 Acceptance-0 | - | Filter9 Acceptance-0 | Filter9 Acceptance-0 |
| 48H | Filter10 Acceptance-3 | - | Filter10 Acceptance-3 | Filter10 Acceptance-3 |
| 49H | Filter10 Acceptance-2 | - | Filter10 Acceptance-2 | Filter10 Acceptance-2 |
| 4AH | Filter10 Acceptance-1 | - | Filter10 Acceptance-1 | Filter10 Acceptance-1 |
| 4BH | Filter10 Acceptance-0 | - | Filter10 Acceptance-0 | Filter10 Acceptance-0 |
| 4CH | Filter11 Acceptance-3 | - | Filter11 Acceptance-3 | Filter11 Acceptance-3 |
| 4DH | Filter11 Acceptance-2 | - | Filter11 Acceptance-2 | Filter11 Acceptance-2 |
| 4EH | Filter11 Acceptance-1 | - | Filter11 Acceptance-1 | Filter11 Acceptance-1 |
| 4FH | Filter11 Acceptance-0 | - | Filter11 Acceptance-0 | Filter11 Acceptance-0 |

| Address (8-bit) | Operating Mode | | Reset Mode | |
|---|---|---|---|---|
| | read | write | read | write |
| 50H | Filter Mask 0-3 | - | Filter Mask 0-3 | Filter Mask 0-3 |
| 51H | Filter Mask 0-2 | - | Filter Mask 0-2 | Filter Mask 0-2 |
| 52H | Filter Mask 0-1 | - | Filter Mask 0-1 | Filter Mask 0-1 |
| 53H | Filter Mask 0-0 | - | Filter Mask 0-0 | Filter Mask 0-0 |
| 54H | Filter Mask 1-3 | - | Filter Mask 1-3 | Filter Mask 1-3 |
| 55H | Filter Mask 1-2 | - | Filter Mask 1-2 | Filter Mask 1-2 |
| 56H | Filter Mask 1-1 | - | Filter Mask 1-1 | Filter Mask 1-1 |
| 57H | Filter Mask 1-0 | - | Filter Mask 1-0 | Filter Mask 1-0 |
| 58H | Filter Mask 2-3 | - | Filter Mask 2-3 | Filter Mask 2-3 |
| 59H | Filter Mask 2-2 | - | Filter Mask 2-2 | Filter Mask 2-2 |
| 5AH | Filter Mask 2-1 | - | Filter Mask 2-1 | Filter Mask 2-1 |
| 5BH | Filter Mask 2-0 | - | Filter Mask 2-0 | Filter Mask 2-0 |
| 5CH | Filter Mask 3-3 | - | Filter Mask 3-3 | Filter Mask 3-3 |
| 5DH | Filter Mask 3-2 | - | Filter Mask 3-2 | Filter Mask 3-2 |
| 5EH | Filter Mask 3-1 | - | Filter Mask 3-1 | Filter Mask 3-1 |
| 5FH | Filter Mask 3-0 | - | Filter Mask 3-0 | Filter Mask 3-0 |
| 60H | Filter Mask 4-3 | - | Filter Mask 4-3 | Filter Mask 4-3 |
| 61H | Filter Mask 4-2 | - | Filter Mask 4-2 | Filter Mask 4-2 |
| 62H | Filter Mask 4-1 | - | Filter Mask 4-1 | Filter Mask 4-1 |
| 63H | Filter Mask 4-0 | - | Filter Mask 4-0 | Filter Mask 4-0 |
| 64H | Filter Mask 5-3 | - | Filter Mask 5-3 | Filter Mask 5-3 |
| 65H | Filter Mask 5-2 | - | Filter Mask 5-2 | Filter Mask 5-2 |
| 66H | Filter Mask 5-1 | - | Filter Mask 5-1 | Filter Mask 5-1 |
| 67H | Filter Mask 5-0 | - | Filter Mask 5-0 | Filter Mask 5-0 |
| 68H | Filter Mask 6-3 | - | Filter Mask 6-3 | Filter Mask 6-3 |
| 69H | Filter Mask 6-2 | - | Filter Mask 6-2 | Filter Mask 6-2 |
| 6AH | Filter Mask 6-1 | - | Filter Mask 6-1 | Filter Mask 6-1 |
| 6BH | Filter Mask 6-0 | - | Filter Mask 6-0 | Filter Mask 6-0 |
| 6CH | Filter Mask 7-3 | - | Filter Mask 7-3 | Filter Mask 7-3 |
| 6DH | Filter Mask 7-2 | - | Filter Mask 7-2 | Filter Mask 7-2 |
| 6EH | Filter Mask 7-1 | - | Filter Mask 7-1 | Filter Mask 7-1 |
| 6FH | Filter Mask 7-0 | - | Filter Mask 7-0 | Filter Mask 7-0 |
| 70H | Filter-Low Enable | - | Filter-Low Enable | Filter-Low Enable |
| 71H | Filter-High Enable | - | Filter-High Enable | Filter-High Enable |
| 72H | Filter Interrupt Enable | Filter Interrupt Enable | Filter Interrupt Enable | Filter Interrupt Enable |
| 73H | Extra Interrupt Enable | Extra Interrupt Enable | Extra Interrupt Enable | Extra Interrupt Enable |
| 74H | Filter Interrupt Flag | - | Filter Interrupt Flag | - |
| 75H | Extra Interrupt Flag | - | Extra Interrupt Flag | - |
| 76H | Transceiver Error Code | Transceiver Error Code | Transceiver Error Code | Transceiver Error Code |
| 77H | Regulator Configuration | Regulator Configuration | Regulator Configuration | Regulator Configuration |
| 78H | I/O Control | I/O Control | I/O Control | I/O Control |
| 79H | GPIO Data | GPIO Data | GPIO Data | GPIO Data |

| Address (8-bit) | Operating Mode | | Reset Mode | |
|---|---|---|---|---|
| | read | write | read | write |
| 7AH | RX Frame Count | - | RX Frame Count | - |
| 7B~7CH | - | - | - | - |
| 7DH | Fault Detect Time | Fault Detect Time | Fault Detect Time | Fault Detect Time |
| 7EH | - | CAN Test Enable | - | CAN Test Enable |
| 7FH | - | Protection Locker | - | Protection Locker |
| 80H | TX0 Configuration | TX0 Configuration | TX0 Configuration | TX0 Configuration |
| 81H | TX0 Information | TX0 Information | TX0 Information | TX0 Information |
| 82H | TX0 ID-3 | TX0 ID-3 | TX0 ID-3 | TX0 ID-3 |
| 83H | TX0 ID-2 | TX0 ID-2 | TX0 ID-2 | TX0 ID-2 |
| 84H | TX0 ID-1 | TX0 ID-1 | TX0 ID-1 | TX0 ID-1 |
| 85H | TX0 ID-0 | TX0 ID-0 | TX0 ID-0 | TX0 ID-0 |
| 86H | TX0 DATA-0 | TX0 DATA-0 | TX0 DATA-0 | TX0 DATA-0 |
| 87H | TX0 DATA-1 | TX0 DATA-1 | TX0 DATA-1 | TX0 DATA-1 |
| 88H | TX0 DATA-2 | TX0 DATA-2 | TX0 DATA-2 | TX0 DATA-2 |
| 89H | TX0 DATA-3 | TX0 DATA-3 | TX0 DATA-3 | TX0 DATA-3 |
| 8AH | TX0 DATA-4 | TX0 DATA-4 | TX0 DATA-4 | TX0 DATA-4 |
| 8BH | TX0 DATA-5 | TX0 DATA-5 | TX0 DATA-5 | TX0 DATA-5 |
| 8CH | TX0 DATA-6 | TX0 DATA-6 | TX0 DATA-6 | TX0 DATA-6 |
| 8DH | TX0 DATA-7 | TX0 DATA-7 | TX0 DATA-7 | TX0 DATA-7 |
| 8E~8FH | - | - | - | - |
| 90H | TX1 Configuration | TX1 Configuration | TX1 Configuration | TX1 Configuration |
| 91H | TX1 Information | TX1 Information | TX1 Information | TX1 Information |
| 92H | TX1 ID-3 | TX1 ID-3 | TX1 ID-3 | TX1 ID-3 |
| 93H | TX1 ID-2 | TX1 ID-2 | TX1 ID-2 | TX1 ID-2 |
| 94H | TX1 ID-1 | TX1 ID-1 | TX1 ID-1 | TX1 ID-1 |
| 95H | TX1 ID-0 | TX1 ID-0 | TX1 ID-0 | TX1 ID-0 |
| 96H | TX1 DATA-0 | TX1 DATA-0 | TX1 DATA-0 | TX1 DATA-0 |
| 97H | TX1 DATA-1 | TX1 DATA-1 | TX1 DATA-1 | TX1 DATA-1 |
| 98H | TX1 DATA-2 | TX1 DATA-2 | TX1 DATA-2 | TX1 DATA-2 |
| 99H | TX1 DATA-3 | TX1 DATA-3 | TX1 DATA-3 | TX1 DATA-3 |
| 9AH | TX1 DATA-4 | TX1 DATA-4 | TX1 DATA-4 | TX1 DATA-4 |
| 9BH | TX1 DATA-5 | TX1 DATA-5 | TX1 DATA-5 | TX1 DATA-5 |
| 9CH | TX1 DATA-6 | TX1 DATA-6 | TX1 DATA-6 | TX1 DATA-6 |
| 9DH | TX1 DATA-7 | TX1 DATA-7 | TX1 DATA-7 | TX1 DATA-7 |
| 9E~9FH | - | - | - | - |
| A0H | TX2 Configuration | TX2 Configuration | TX2 Configuration | TX2 Configuration |
| A1H | TX2 Information | TX2 Information | TX2 Information | TX2 Information |
| A2H | TX2 ID-3 | TX2 ID-3 | TX2 ID-3 | TX2 ID-3 |
| A3H | TX2 ID-2 | TX2 ID-2 | TX2 ID-2 | TX2 ID-2 |
| A4H | TX2 ID-1 | TX2 ID-1 | TX2 ID-1 | TX2 ID-1 |
| A5H | TX2 ID-0 | TX2 ID-0 | TX2 ID-0 | TX2 ID-0 |
| A6H | TX2 DATA-0 | TX2 DATA-0 | TX2 DATA-0 | TX2 DATA-0 |

| Address (8-bit) | Operating Mode | | Reset Mode | |
|---|---|---|---|---|
| | read | write | read | write |
| A7H | TX2 DATA-1 | TX2 DATA-1 | TX2 DATA-1 | TX2 DATA-1 |
| A8H | TX2 DATA-2 | TX2 DATA-2 | TX2 DATA-2 | TX2 DATA-2 |
| A9H | TX2 DATA-3 | TX2 DATA-3 | TX2 DATA-3 | TX2 DATA-3 |
| AAH | TX2 DATA-4 | TX2 DATA-4 | TX2 DATA-4 | TX2 DATA-4 |
| ABH | TX2 DATA-5 | TX2 DATA-5 | TX2 DATA-5 | TX2 DATA-5 |
| ACH | TX2 DATA-6 | TX2 DATA-6 | TX2 DATA-6 | TX2 DATA-6 |
| ADH | TX2 DATA-7 | TX2 DATA-7 | TX2 DATA-7 | TX2 DATA-7 |
| AE~AFH | - | - | - | - |
| B0H | TX3 Configuration | TX3 Configuration | TX3 Configuration | TX3 Configuration |
| B1H | TX3 Information | TX3 Information | TX3 Information | TX3 Information |
| B2H | TX3 ID-3 | TX3 ID-3 | TX3 ID-3 | TX3 ID-3 |
| B3H | TX3 ID-2 | TX3 ID-2 | TX3 ID-2 | TX3 ID-2 |
| B4H | TX3 ID-1 | TX3 ID-1 | TX3 ID-1 | TX3 ID-1 |
| B5H | TX3 ID-0 | TX3 ID-0 | TX3 ID-0 | TX3 ID-0 |
| B6H | TX3 DATA-0 | TX3 DATA-0 | TX3 DATA-0 | TX3 DATA-0 |
| B7H | TX3 DATA-1 | TX3 DATA-1 | TX3 DATA-1 | TX3 DATA-1 |
| B8H | TX3 DATA-2 | TX3 DATA-2 | TX3 DATA-2 | TX3 DATA-2 |
| B9H | TX3 DATA-3 | TX3 DATA-3 | TX3 DATA-3 | TX3 DATA-3 |
| BAH | TX3 DATA-4 | TX3 DATA-4 | TX3 DATA-4 | TX3 DATA-4 |
| BBH | TX3 DATA-5 | TX3 DATA-5 | TX3 DATA-5 | TX3 DATA-5 |
| BCH | TX3 DATA-6 | TX3 DATA-6 | TX3 DATA-6 | TX3 DATA-6 |
| BDH | TX3 DATA-7 | TX3 DATA-7 | TX3 DATA-7 | TX3 DATA-7 |
| BE~BFH | - | - | - | - |
| C0H | TX4 Configuration | TX4 Configuration | TX4 Configuration | TX4 Configuration |
| C1H | TX4 Information | TX4 Information | TX4 Information | TX4 Information |
| C2H | TX4 ID-3 | TX4 ID-3 | TX4 ID-3 | TX4 ID-3 |
| C3H | TX4 ID-2 | TX4 ID-2 | TX4 ID-2 | TX4 ID-2 |
| C4H | TX4 ID-1 | TX4 ID-1 | TX4 ID-1 | TX4 ID-1 |
| C5H | TX4 ID-0 | TX4 ID-0 | TX4 ID-0 | TX4 ID-0 |
| C6H | TX4 DATA-0 | TX4 DATA-0 | TX4 DATA-0 | TX4 DATA-0 |
| C7H | TX4 DATA-1 | TX4 DATA-1 | TX4 DATA-1 | TX4 DATA-1 |
| C8H | TX4 DATA-2 | TX4 DATA-2 | TX4 DATA-2 | TX4 DATA-2 |
| C9H | TX4 DATA-3 | TX4 DATA-3 | TX4 DATA-3 | TX4 DATA-3 |
| CAH | TX4 DATA-4 | TX4 DATA-4 | TX4 DATA-4 | TX4 DATA-4 |
| CBH | TX4 DATA-5 | TX4 DATA-5 | TX4 DATA-5 | TX4 DATA-5 |
| CCH | TX4 DATA-6 | TX4 DATA-6 | TX4 DATA-6 | TX4 DATA-6 |
| CDH | TX4 DATA-7 | TX4 DATA-7 | TX4 DATA-7 | TX4 DATA-7 |
| CE~CFH | - | - | - | - |
| D0H | TX5 Configuration | TX5 Configuration | TX5 Configuration | TX5 Configuration |
| D1H | TX5 Information | TX5 Information | TX5 Information | TX5 Information |
| D2H | TX5 ID-3 | TX5 ID-3 | TX5 ID-3 | TX5 ID-3 |
| D3H | TX5 ID-2 | TX5 ID-2 | TX5 ID-2 | TX5 ID-2 |

| Address (8-bit) | Operating Mode | | Reset Mode | |
|---|---|---|---|---|
| | read | write | read | write |
| D4H | TX5 ID-1 | TX5 ID-1 | TX5 ID-1 | TX5 ID-1 |
| D5H | TX5 ID-0 | TX5 ID-0 | TX5 ID-0 | TX5 ID-0 |
| D6H | TX5 DATA-0 | TX5 DATA-0 | TX5 DATA-0 | TX5 DATA-0 |
| D7H | TX5 DATA-1 | TX5 DATA-1 | TX5 DATA-1 | TX5 DATA-1 |
| D8H | TX5 DATA-2 | TX5 DATA-2 | TX5 DATA-2 | TX5 DATA-2 |
| D9H | TX5 DATA-3 | TX5 DATA-3 | TX5 DATA-3 | TX5 DATA-3 |
| DAH | TX5 DATA-4 | TX5 DATA-4 | TX5 DATA-4 | TX5 DATA-4 |
| DBH | TX5 DATA-5 | TX5 DATA-5 | TX5 DATA-5 | TX5 DATA-5 |
| DCH | TX5 DATA-6 | TX5 DATA-6 | TX5 DATA-6 | TX5 DATA-6 |
| DDH | TX5 DATA-7 | TX5 DATA-7 | TX5 DATA-7 | TX5 DATA-7 |
| DE~DFH | - | - | - | - |
| E0H | TX6 Configuration | TX6 Configuration | TX6 Configuration | TX6 Configuration |
| E1H | TX6 Information | TX6 Information | TX6 Information | TX6 Information |
| E2H | TX6 ID-3 | TX6 ID-3 | TX6 ID-3 | TX6 ID-3 |
| E3H | TX6 ID-2 | TX6 ID-2 | TX6 ID-2 | TX6 ID-2 |
| E4H | TX6 ID-1 | TX6 ID-1 | TX6 ID-1 | TX6 ID-1 |
| E5H | TX6 ID-0 | TX6 ID-0 | TX6 ID-0 | TX6 ID-0 |
| E6H | TX6 DATA-0 | TX6 DATA-0 | TX6 DATA-0 | TX6 DATA-0 |
| E7H | TX6 DATA-1 | TX6 DATA-1 | TX6 DATA-1 | TX6 DATA-1 |
| E8H | TX6 DATA-2 | TX6 DATA-2 | TX6 DATA-2 | TX6 DATA-2 |
| E9H | TX6 DATA-3 | TX6 DATA-3 | TX6 DATA-3 | TX6 DATA-3 |
| EAH | TX6 DATA-4 | TX6 DATA-4 | TX6 DATA-4 | TX6 DATA-4 |
| EBH | TX6 DATA-5 | TX6 DATA-5 | TX6 DATA-5 | TX6 DATA-5 |
| ECH | TX6 DATA-6 | TX6 DATA-6 | TX6 DATA-6 | TX6 DATA-6 |
| EDH | TX6 DATA-7 | TX6 DATA-7 | TX6 DATA-7 | TX6 DATA-7 |
| EE~EFH | - | - | - | - |
| F0H | TX7 Configuration | TX7 Configuration | TX7 Configuration | TX7 Configuration |
| F1H | TX7 Information | TX7 Information | TX7 Information | TX7 Information |
| F2H | TX7 ID-3 | TX7 ID-3 | TX7 ID-3 | TX7 ID-3 |
| F3H | TX7 ID-2 | TX7 ID-2 | TX7 ID-2 | TX7 ID-2 |
| F4H | TX7 ID-1 | TX7 ID-1 | TX7 ID-1 | TX7 ID-1 |
| F5H | TX7 ID-0 | TX7 ID-0 | TX7 ID-0 | TX7 ID-0 |
| F6H | TX7 DATA-0 | TX7 DATA-0 | TX7 DATA-0 | TX7 DATA-0 |
| F7H | TX7 DATA-1 | TX7 DATA-1 | TX7 DATA-1 | TX7 DATA-1 |
| F8H | TX7 DATA-2 | TX7 DATA-2 | TX7 DATA-2 | TX7 DATA-2 |
| F9H | TX7 DATA-3 | TX7 DATA-3 | TX7 DATA-3 | TX7 DATA-3 |
| FAH | TX7 DATA-4 | TX7 DATA-4 | TX7 DATA-4 | TX7 DATA-4 |
| FBH | TX7 DATA-5 | TX7 DATA-5 | TX7 DATA-5 | TX7 DATA-5 |
| FCH | TX7 DATA-6 | TX7 DATA-6 | TX7 DATA-6 | TX7 DATA-6 |
| FDH | TX7 DATA-7 | TX7 DATA-7 | TX7 DATA-7 | TX7 DATA-7 |
| FE~FFH | - | - | - | - |

## 1.4 Configuration, Control and Status Registers

**CANMODE (0x00) CAN Mode Configuration Register R/W (0x41)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | TSE | TSL | - | - | ERAR | STE | LOM | RSTM |
| WR | TSE | TSL | - | - | ERAR | STE | LOM | RSTM |

TSE — Transceiver Enable
TSE=1 will enable the embedded CAN transceiver
TSE=0 will disable the embedded CAN transceiver
TSE can be read and written in Reset Mode and is read-only in Operating mode.

TSL — Transceiver Slope
If TSL=1, transceiver's output buffer at normal speed
If TSL=0, transceiver's output buffer at low speed
TSL can be read and written in Reset Mode and is read-only in Operating mode.

ERAR — Error Auto Recovery
ERAR=1 will allow auto recovery from bus off state
When this bit is set, auto recovery from Bus Off state to Error Active state is enabled. The auto recovery condition is met when 128 occurrences of bus-free time (11 consecutive 'recessive' bits) is received. When exiting the Bus Off state, TEC and REC are cleared to 0 by hardware if ERAR=1.
ERAR=0 for disable the auto recovery function.
If ERAR=0, software must clear REC and TEC to exit of Bus Off state to Error Active state. Please note in the entry of Bus Off state, if ERAR=0, RSTM will be set by hardware, so modification of REC and TEC is allowed. After clearing REC and TEC, the software must also clear RSTM bit to return to normal operation.

STE — Self-Test Mode
STM=1 enables self-test mode.
When STM=1, the only difference from normal operation mode is that a message is treated as successful transmitted by ignoring the ACK error.
STM=0 for normal mode operation

LOM — Listen-Only Mode
LOM=1 will set the CAN controller as listen-only mode
When LOM=1, CAN controller will work in Listen-Only mode. In this mode, the CAN controller will only perform receiver functions and does not engage any transmission (including ACK signaling). All other functions can be used like in Operating mode. In this mode, the error counters are stopped at the current value and message transmission is not possible. But when detecting bus error, listen-only node will still transmit error frame.
LOM=0 for normal mode operation

RSTM — Reset Mode
RSTM=1 will force the CAN Controller into rest mode regardless of other setting, and will initialize all state machine. It does not affect the register contents. RSTM defaults to 1 after system reset. RSTM is set to 1 by hardware when entering into Bus-Off state if ERAR=0.
RSTM must be cleared to 0 to allow the CAN Controller into normal operation.
Please note RSTM must be set to 1 first in order to modify TSE, TSL, ERAR, STE, and LOM. The write cycle of RSTM must be preceding the modification cycles of TSE, TSL, ERAR, STE, and LOM.

**CANCMD (0x01) CAN Command Register WO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | - | - | - | - | - | - | - |
| WR | - | - | OFR | SRR | - | - | CDO | RRB |

This register is write-only and reading will return 0x00.

OFR — Overload Frame Request
OFR=1 will request transmitting an overload frame. The overload frame is transmitted the first bit time of next expected INTERMISSION. This bit is self-cleared by hardware after the overload frame is transmitted. Since the overload frame will be transmitted regardless of the bus condition, no interrupt is generated at the completion.

SRR — Self Receive Request

SRR=1 will result a message be transmitted and received simultaneously. Upon self-reception request command, a message is transmitted and simultaneously received if the acceptance filter is set to the corresponding identifier. This command is used mainly for self-test of the controller. If STE=1 which enable the self-test mode, the transmitter and receiver will ignore ACK error, then transmit and receive can be successful without other nodes present. This command is used mainly for self-test of the controller. This bit is self- cleared by hardware when transmit and receive operations are completed and both transmit and receive interrupts will be generated. SRR only influences transmit buffer 0.

CDO — Clear Data Overrun

CDO=1 will clear data overrun status bit. This command bit is used to clear the data overrun condition indicated by the data overrun status bit. This command bit is self-cleared by hardware.

RRB — Release Receive Buffer

After reading the contents of the receive buffer at 0X11~0X1D, the CPU can release this memory space in the RXFIFO by setting RRB to logic 1. This may result in another message becoming immediately available within the receive buffer. If there is no other message available, the receive interrupt bit (RI) is reset. This command bit is self-cleared by hardware.

**CANSTAT (0x02) CAN Status Register RO (0x08)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | BOS | ES | TS | RS | TCS | BEPS | DOS | RBS |
| WR | - | - | - | - | - | - | - | - |

This register is read-only, corresponding bits are self-cleared by hardware when the condition is not met.

BOS — Bus Off State Status

BOS=1 indicates the CAN controller is in the Bus Off State.

BOS=0 indicates the CAN controller is involved in bus activities.

Bus Off state is defined in the error handling state diagram. Possible exit of Bus Off state is by Bus Auto recovery, or by writing a value less than 255 into TEC under reset mode.

ES — Error Status

ES=1 indicates at least one of the error counters has reached or exceeded the warning limit defined by the Error Warning Limit register.

ES=0 indicates warning limits are not exceeded.

Errors detected during reception or transmission will affect the error counters according to the CAN 2.0B protocol specification. ES is set when at least one of the error counters has reached or exceeded the warning limit setting (EWLR). An error warning interrupt is generated, if enabled. The default value of EWLR after hardware reset is 96.

TS — Transmit Status

TS=1 indicates CAN controller is transmitting a message.

RS — Receive Status

RS=1 indicates CAN controller is receiving a message.

If both TS and RS are 0, the CAN controller is idle.

TCS — Transmit Complete Status

TCS=1 indicates last requested message transmission has been successfully completed

TCR is cleared to 0 when any transmit request command or SRR command is issued. And TCS is set to 1 by hardware when all messages are sent successfully or aborted by setting AT. TCR remains to be 0 during the message transmission, and TCS is set to 1 by hardware if all messages are sent successfully or aborted.

BEPS — Bus Error Passive Status

BEPS=1 indicates the CAN controller enter Error Passive mode when either Transmit Error Counter or Receive Error Counter equal or exceeds 128..

BEPS=0 indicates the CAN controller leave Error Passive mode when return Error Active mode or enter Bus off state.

DOS — Data Overrun Status

DOS=1 indicates that a message was lost because there was not enough memory space for that message in the RXFIFO. And an overrun interrupt is generated with DOI flag set.

DOS is cleared by setting the Clear Data Overrun (CDO) command bit to logic 1 or reset mode = 1.

RBS — Receive Buffer Status. RBS=1 indicates one or more valid messages are available in RXFIFO. RBS is self-cleared by hardware when RXFIFO is empty.

**CANINTF (0x03) CAN Interrupt Flag Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | BEI | ALI | EPI | BOI | DOI | EI | TI | RI |
| WR | - | - | - | - | - | - | - | - |

This is a read-only register. All flags are event change triggering so only the change of the condition will cause the interrupt and corresponding flag to be set. All bits are cleared after read except RI.

BEI — Bus Error Interrupt Flag
BEI is set when the CAN controller detects error(s) on the CAN-bus. The error types include format error, bit stuff error, bit error (when data or CRC transmitted does not match with received), CRC error, and ACK error.

ALI — Arbitration Lost Interrupt
ALI is set when a message was not sent successfully due to loss of arbitration.

EPI — Error Passive Interrupt
EPI is set when the CAN controller enters the error passive state, or the CAN controller is already in the error passive state and enters the error active state.

BOI — Bus Off State Interrupt
BOI is set when the CAN controller in the bus off state or equivalently BOS=1. Since BOI and BOS is 1 whenever in Bus Off state, the software must take care to disable BOIE when entering BOI ISR and then correct the BOS status otherwise the interrupt will occur indefinitely. BOI is cleared when read.

DOI — Data Overrun Interrupt
DOI is set on a '0-to-1' transition of the data overrun status (DOS). DOI is an interrupt to indicate that a message was lost because there was not enough memory space for that message in the RXFIFO. DOI is cleared when read.

EI — Error Warning Interrupt
EI is set on when REC or TEC is greater or equal to the CANEWLR setting or equivalently when ES changes from 0 to 1. EI is cleared when read.

TI — Transmit Interrupt
TI is set whenever the transmit buffer becomes free or equivalently the transmission of the message has finished and TBS changes from 0 to 1. TI is cleared when read.

RI — Receive Interrupt
RI is set when a new message has been put into the RXFIFO by the CAN controller. RI is cleared when all messages in RXFIFO have been read and released by RRB command.

**CANIE (0x04) CAN Interrupt Enable Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | BEIE | ALIE | EPIE | BOIE | DOIE | EIE | TIE | RIE |
| WR | BEIE | ALIE | EPIE | BOIE | DOIE | EIE | TIE | RIE |

The interrupt enable register allows indicating different types of interrupt source. The register appears to the CPU as a read/write memory.

BEIE — Bus Error Interrupt Enable
ALIE — Arbitration Lost Interrupt Enable
EPIE — Error Passive Interrupt Enable
BOIE — Bus Off Interrupt Enable
DOIE — Data Overrun Interrupt Enable
EIE — Error Warning Interrupt Enable
TIE — Transmit Interrupt Enable
RIE — Receive Interrupt Enable

**CANBTR0 (0x09) CAN Bus Timing Register 0 R/W (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | SJW[1-0] | | CANCS[5-0] | | | | | |
| WR | SJW[1-0] | | CANCS[5-0] | | | | | |

CANBR[5-0] CANBR[5-0] defines the CAN Controller Clock
The CAN controller clock CANCLK = SYSCLK/2/(CANCS[5-0]+1). This defines the CAN Time Quanta from the system clock.

SJW[1-0] Synchronization Jump Length
This defines the maximum adjustment in CANCLK period units for re-synchronization. The receiver uses the falling edges to determine the physical delay of the network. The synchronization is achieved by inserting a delay compensation segment in the bit time after SYNC SEG and thus also delays the sampling points of the local receiver. SJW[1-0]+1 is used to set the maximum allowed delay segment in CANCLK period.

**CANBTR1 (0x0A) CAN Bus Timing Register 1 R/W (0x67)**

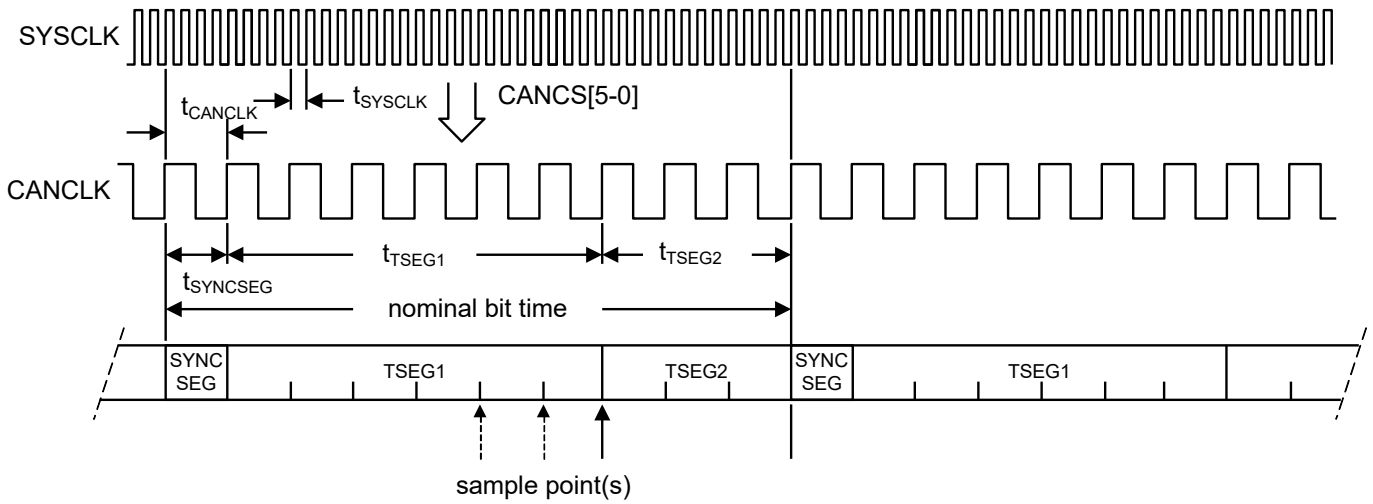
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | SAM | TSEG2[2-0] | | | TSEG1[3-0] | | | |
| WR | SAM | TSEG2[2-0] | | | TSEG1[3-0] | | | |

SAM SAM=1 will use triple-sampling method. The triple-sampling method uses three instances between TSEG1 and TSEG2 as shown in the following timing diagram. The data is determined by the majority of the sampled results.
SAM=0 will use single-sampling method. The single sampling occurs at the time instance between TSEG1 and TSEG2

TSEG2[2-0] TSEG2[2-0] defines the bit time component 2. TSEG2 = TSEG2[2-0] + 1.

TSEG1[3-0] TSEG1[3-0] defines the bit time component 1. TSEG1 = TSEG1[3-0] + 1.

A CAN bit time is partitioned by CANCLK periods as the following diagram. The first CAN CLK period is used for edge synchronization, and TSEG1 is used for defining the sampling edge. For SAM=0, the sampling edge is the first edge of TSEG2. For SAM=1, the sampling edges are the last two edges of TSEG1, and the first edge of TSEG2. For typical CAN bus design, TSEG1 and TSEG2 is organized for sampling at the last two quanta (period of CANCLK).




CAN baud rate is thus can be calculated as CANCLK/((TSEG2[2-0]+1)+(TSEG1[3-0]+1)+1). Or in terms of SYSCLK, SYSCLK/2/(CANCS[5-0]+1)/{(TSEG2[2-0]+1)+(TSEG1[3-0]+1)+1}. Also note, with re-synchronization, a delay compensation segment is inserted between SYNC SEG and TSEG1. The maximum length of delay compensation segment is limited by SJW[1-0].

**CANALC (0x07) CAN Arbitration Lost Capture Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | - | - | ALC4 | ALC3 | ALC2 | ALC1 | ALC0 |
| WR | - | - | - | - | - | - | - | - |

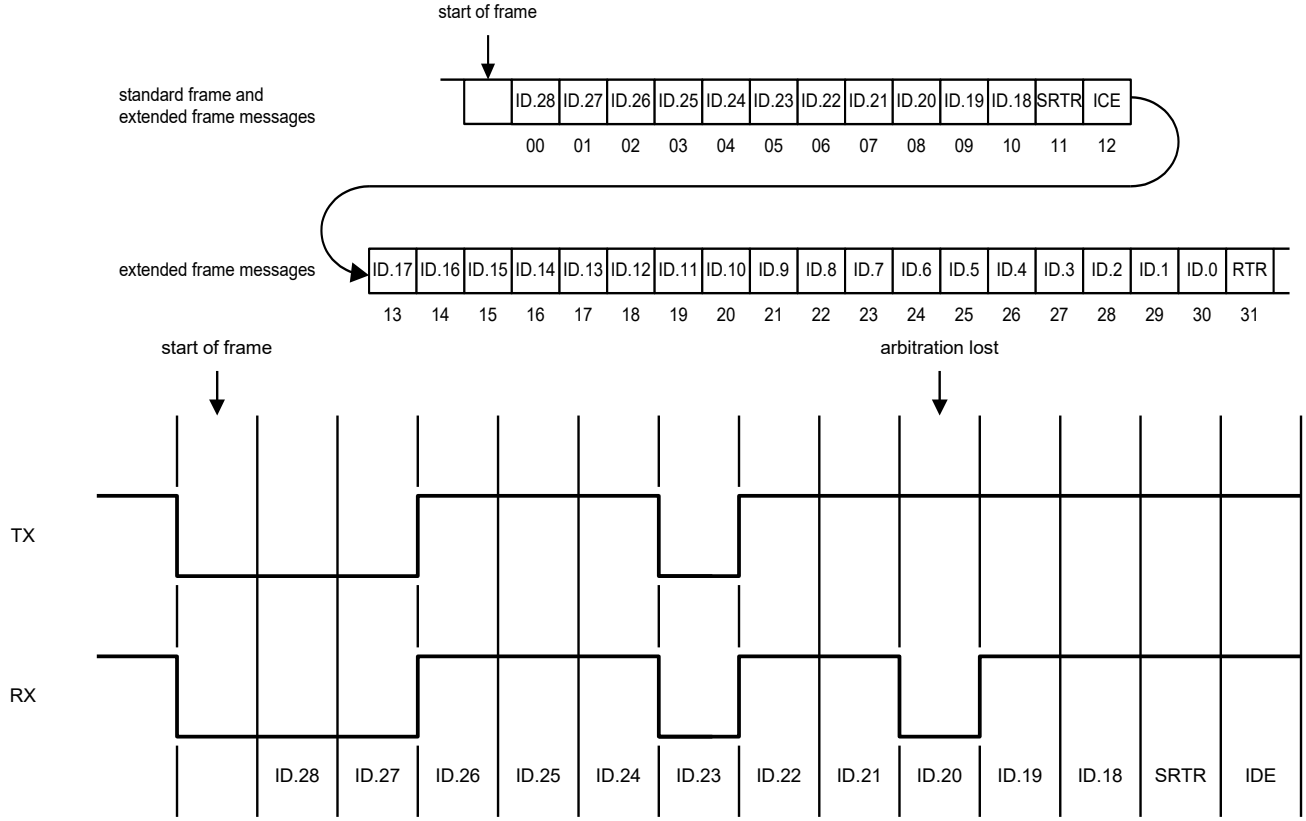
This register is read only and reports the bit locations of a lost arbitration when transmitting a message. The following table defines the corresponding relationship.

| Bits | | | | | Decimal Value | Description |
|---|---|---|---|---|---|---|
| ALC4 | ALC3 | ALC2 | ALC1 | ALC0 | | |
| 0 | 0 | 0 | 0 | 0 | 0 | Arbitration lost in bit ID28 |
| 0 | 0 | 0 | 0 | 1 | 1 | Arbitration lost in bit ID27 |
| 0 | 0 | 0 | 1 | 0 | 2 | Arbitration lost in bit ID26 |
| 0 | 0 | 0 | 1 | 1 | 3 | Arbitration lost in bit ID25 |
| 0 | 0 | 1 | 0 | 0 | 4 | Arbitration lost in bit ID24 |
| 0 | 0 | 1 | 0 | 1 | 5 | Arbitration lost in bit ID23 |
| 0 | 0 | 1 | 1 | 0 | 6 | Arbitration lost in bit ID22 |
| 0 | 0 | 1 | 1 | 1 | 7 | Arbitration lost in bit ID21 |
| 0 | 1 | 0 | 0 | 0 | 8 | Arbitration lost in bit ID20 |
| 0 | 1 | 0 | 0 | 1 | 9 | Arbitration lost in bit ID19 |
| 0 | 1 | 0 | 1 | 0 | 10 | Arbitration lost in bit ID18 |
| 0 | 1 | 0 | 1 | 1 | 11 | Arbitration lost in bit SRTR(Note 1) |
| 0 | 1 | 1 | 0 | 0 | 12 | Arbitration lost in bit IDE |
| 0 | 1 | 1 | 0 | 1 | 13 | Arbitration lost in bit ID17(Note 2) |
| 0 | 1 | 1 | 1 | 0 | 14 | Arbitration lost in bit ID16(Note 2) |
| 0 | 1 | 1 | 1 | 1 | 15 | Arbitration lost in bit ID15(Note 2) |
| 1 | 0 | 0 | 0 | 0 | 16 | Arbitration lost in bit ID14(Note 2) |
| 1 | 0 | 0 | 0 | 1 | 17 | Arbitration lost in bit ID13(Note 2) |
| 1 | 0 | 0 | 1 | 0 | 18 | Arbitration lost in bit ID12(Note 2) |
| 1 | 0 | 0 | 1 | 1 | 19 | Arbitration lost in bit ID11(Note 2) |
| 1 | 0 | 1 | 0 | 0 | 20 | Arbitration lost in bit ID10(Note 2) |
| 1 | 0 | 1 | 0 | 1 | 21 | Arbitration lost in bit ID9(Note 2) |
| 1 | 0 | 1 | 1 | 0 | 22 | Arbitration lost in bit ID8(Note 2) |
| 1 | 0 | 1 | 1 | 1 | 23 | Arbitration lost in bit ID7(Note 2) |
| 1 | 1 | 0 | 0 | 0 | 24 | Arbitration lost in bit ID6(Note 2) |
| 1 | 1 | 0 | 0 | 1 | 25 | Arbitration lost in bit ID5(Note 2) |
| 1 | 1 | 0 | 1 | 0 | 26 | Arbitration lost in bit ID4(Note 2) |
| 1 | 1 | 0 | 1 | 1 | 27 | Arbitration lost in bit ID3(Note 2) |
| 1 | 1 | 1 | 0 | 0 | 28 | Arbitration lost in bit ID2(Note 2) |
| 1 | 1 | 1 | 0 | 1 | 29 | Arbitration lost in bit ID1(Note 2) |
| 1 | 1 | 1 | 1 | 0 | 30 | Arbitration lost in bit ID0(Note 2) |
| 1 | 1 | 1 | 1 | 1 | 31 | Arbitration lost in bit RTR(Note 2) |

*Note 1: Bit RTR for standard frame messages.*

*Note 2: Extended frame messages only.*

On arbitration lost, the corresponding arbitration lost interrupt is generated. At the same time, the current bit position of the bit stream processor is captured into the Arbitration Lost Capture register. The content in this register is locked until the software read out its contents. The capture mechanism is then activated again. The corresponding interrupt flag located in the interrupt register is cleared during the read access to the interrupt flag register. A new arbitration lost interrupt is not possible until the arbitration lost capture register is read out. The following diagram shows the bit locations of the arbitration lost. And an ALC=0x08 is shown as an example, where arbitration lost is detected when outputting a recessive bit but receiving a dominant bit at ID20.




**CANECC (0x08) CAN Error Code Capture Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | ERRC[1] | ERRC[0] | DIR | SEG[4] | SEG[3] | SEG[2] | SEG[1] | SEG[0] |
| WR | - | - | - | - | - | - | - | - |

This register is read only and cleared after read. This register always keeps the last error status on CAN bus.

ERRC[1-0] Error Code

This reflects the error type as defined in the following table

| ERCC[1] | ERRC[0] | Descriptions |
|---|---|---|
| 0 | 0 | Bit error |
| 0 | 1 | Form error |
| 1 | 0 | Stuff error |
| 1 | 1 | Other type of error |

DIR Direction of Transfer Error

DIR=1 means errors occurred during reception

DIR=0 means errors occurred during transmission

SEG[4-0] Error Locations

| SEG[4-0] | | | | | Function |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | Start of frame |
| 0 | 0 | 0 | 1 | 0 | ID28 to ID21 |
| 0 | 0 | 1 | 1 | 0 | ID20 toID18 |
| 0 | 0 | 1 | 0 | 0 | Bit SRTR |
| 0 | 0 | 1 | 0 | 1 | Bit IDE |
| 0 | 0 | 1 | 1 | 1 | ID17 to ID13 |
| 0 | 1 | 1 | 1 | 1 | ID12 to ID5 |
| 0 | 1 | 1 | 1 | 0 | ID4 to ID0 |
| 0 | 1 | 1 | 0 | 0 | Bit RTR |
| 0 | 1 | 1 | 0 | 1 | Reserved bit 1 |
| 0 | 1 | 0 | 0 | 1 | Reserved bit 0 |
| 0 | 1 | 0 | 1 | 1 | Data length code |
| 0 | 1 | 0 | 1 | 0 | Data field |
| 0 | 1 | 0 | 0 | 0 | CRC sequence |
| 1 | 1 | 0 | 0 | 0 | CRC delimiter |
| 1 | 1 | 0 | 0 | 1 | Acknowledge slot |
| 1 | 1 | 0 | 1 | 1 | Acknowledge delimiter |
| 1 | 1 | 0 | 1 | 0 | End of frame |
| 1 | 0 | 0 | 1 | 0 | Intermission |
| 1 | 0 | 0 | 0 | 1 | Active error flag |
| 1 | 0 | 1 | 1 | 0 | Passive error flag |
| 1 | 0 | 0 | 1 | 1 | Tolerate dominant bits (Note) |
| 1 | 0 | 1 | 1 | 1 | Error delimiter |
| 1 | 1 | 1 | 0 | 0 | Overload flag |

*Note: Any node tolerates up to 7 consecutive 'dominant' bits after send Active Error Flag, Passive Error Flag or Overload Flag.*

If a bus error occurs, the corresponding bus error interrupt is generated. At the same time, the current position of the bit stream processor is captured into the Error Code Capture register. The content in this register is locked until the software read out its content. A new bus error capture is prohibited until the capture register is read. The capture mechanism is then activated again. The corresponding interrupt flag located in the interrupt flag register is cleared by reading of the interrupt flag register.

**CANEWLR (0x0B) CAN Error Warning Limit Register RW (0x60)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANEWLR[7-0] | | | | | | | |
| WR | CANEWLR[7-0] | | | | | | | |

This register is modifiable on in reset mode and appears as read-only in normal operating mode. This register set the warning limit for the error counter. When CANREC or CANTEC value exceed CANEWLR value, an error warning interrupt is generated, and the EI flag is set in the interrupt flag register. CANEWLF defaults to 0x60 after hardware reset.

**CANREC (0x0C) CAN Receive Error Count Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANREC[7-0] | | | | | | | |
| WR | CANREC[7-0] | | | | | | | |

This register is modifiable on in reset mode and appears as read-only in normal operating mode. CANREC contains the current error count. When the controller enters bus-off state, CANREC is re-initialized as 0, and during bus-off, the writing to CANREC has no effect.

**CANTEC (0x0D) CAN Transmit Error Count Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTEC[7-0] | | | | | | | |
| WR | CANTEC[7-0] | | | | | | | |

This register is modifiable on in reset mode and appears as read-only in normal operating mode. CANTEC contains the current error count, and is initialized as 0 after hardware reset. If a bus-off event occurs, the transmit error counter is initialized to 128 is auto-recovery is not enable. The software can just use the reset mode to allow recovery to the normal error-active state.

## 1.5 Transmit Buffer

There are eight independent transmit buffers in IS31IO8972. The global layout of each transmit buffer is shown in the table below. The transmit buffer contains one configuration register and 13-bytes of memory space can be in the form of Standard Frame Format (SFF) or Extended Frame Format (EFF) configuration.

| Address | | Standard Format Frame | Extended Format Frame |
|---|---|---|---|
| 80H / 90H / A0H / B0H / C0H / D0H / E0H / F0H | : | TXn Configuration | TXn Configuration |
| 81H / 91H / A1H / B1H / C1H / D1H / E1H / F1H | : | TXn Frame Information | TXn Frame Information |
| 82H / 92H / A2H / B2H / C2H / D2H / E2H / F2H | : | Reserved | TXn Identifier 3 |
| 83H / 93H / A3H / B3H / C3H / D3H / E3H / F3H | : | Reserved | TXn Identifier 2 |
| 84H / 94H / A4H / B4H / C4H / D4H / E4H / F4H | : | TXn Identifier 1 | TXn Identifier 1 |
| 85H / 95H / A5H / B5H / C5H / D5H / E5H / F5H | : | TXn Identifier 0 | TXn Identifier 0 |
| 86H / 96H / A6H / B6H / C6H / D6H / E6H / F6H | : | TXn Data 0 | TXn Data 0 |
| 87H / 97H / A7H / B7H / C7H / D7H / E7H / F7H | : | TXn Data 1 | TXn Data 1 |
| 88H / 98H / A8H / B8H / C8H / D8H / E8H / F8H | : | TXn Data 2 | TXn Data 2 |
| 89H / 99H / A9H / B9H / C9H / D9H / E9H / F9H | : | TXn Data 3 | TXn Data 3 |
| 8AH / 9AH / AAH / BAH / CAH / DAH / EAH / FAH | : | TXn Data 4 | TXn Data 4 |
| 8BH / 9BH / ABH / BBH / CBH / DBH / EBH / FBH | : | TXn Data 5 | TXn Data 5 |
| 8CH / 9CH / ACH / BCH / CCH / DCH / ECH / FCH | : | TXn Data 6 | TXn Data 6 |
| 8DH / 9DH / ADH / BDH / CDH / DDH / EDH / FDH | : | TXn Data 7 | TXn Data 7 |

The transmit buffer 0 has the highest priority to issue a transmission if bus idle. And the transmit buffer 7 has the lowest priority if more than one transmit requests were set at the same time. Following table is the address location of transmit buffer 0 to transmit buffer 7.

| Transmit Buffer | Name | Description | Address |
|---|---|---|---|
| Transmit Buffer 0 | CANT0CFG | CAN Transmit 0 Configuration Register | 0x80 |
| | CANT0FIR | CAN Transmit 0 Frame Information Register | 0x81 |
| | CANTX0ID3-0 | CAN Transmit 0 Identifier Register 3~0 | 0x82 ~ 0x85 |
| | CANTX0DATA1-8 | CAN Transmit 0 Data Register 1~8 | 0x86 ~ 0x8D |
| Transmit Buffer 1 | CANT1CFG | CAN Transmit 1 Configuration Register | 0x90 |
| | CANT1FIR | CAN Transmit 1 Frame Information Register | 0x91 |

| | | | |
|---|---|---|---|
| | CANTX1ID3-0 | CAN Transmit 1 Identifier Register 3~0 | 0x92 ~ 0x95 |
| | CANTX1DATA1-8 | CAN Transmit 1 Data Register 1~8 | 0x96 ~ 0x9D |
| Transmit Buffer 2 | CANT2CFG | CAN Transmit 2 Configuration Register | 0xA0 |
| | CANT2FIR | CAN Transmit 2 Frame Information Register | 0xA1 |
| | CANTX2ID3-0 | CAN Transmit 2 Identifier Register 3~0 | 0xA2 ~ 0xA5 |
| | CANTX2DATA1-8 | CAN Transmit 2 Data Register 1~8 | 0xA6 ~ 0xAD |
| Transmit Buffer 3 | CANT3CFG | CAN Transmit 3 Configuration Register | 0xB0 |
| | CANT3FIR | CAN Transmit 3 Frame Information Register | 0xB1 |
| | CANTX3ID3-0 | CAN Transmit 3 Identifier Register 3~0 | 0xB2 ~ 0xB5 |
| | CANTX3DATA1-8 | CAN Transmit 3 Data Register 1~8 | 0xB6 ~ 0xBD |
| Transmit Buffer 4 | CANT4CFG | CAN Transmit 4 Configuration Register | 0xC0 |
| | CANT4FIR | CAN Transmit 4 Frame Information Register | 0xC1 |
| | CANTX4ID3-0 | CAN Transmit 4 Identifier Register 3~0 | 0xC2 ~ 0xC5 |
| | CANTX4DATA1-8 | CAN Transmit 4 Data Register 1~8 | 0xC6 ~ 0xCD |
| Transmit Buffer 5 | CANT5CFG | CAN Transmit 5 Configuration Register | 0xD0 |
| | CANT5FIR | CAN Transmit 5 Frame Information Register | 0xD1 |
| | CANTX5ID3-0 | CAN Transmit 5 Identifier Register 3~0 | 0xD2 ~ 0xD5 |
| | CANTX5DATA1-8 | CAN Transmit 5 Data Register 1~8 | 0xD6 ~ 0xDD |
| Transmit Buffer 6 | CANT6CFG | CAN Transmit 6 Configuration Register | 0xE0 |
| | CANT6FIR | CAN Transmit 6 Frame Information Register | 0xE1 |
| | CANTX6ID3-0 | CAN Transmit 6 Identifier Register 3~0 | 0xE2 ~ 0xE5 |
| | CANTX6DATA1-8 | CAN Transmit 6 Data Register 1~8 | 0xE6 ~ 0xED |
| Transmit Buffer 7 | CANT7CFG | CAN Transmit 7 Configuration Register | 0xF0 |
| | CANT7FIR | CAN Transmit 7 Frame Information Register | 0xF1 |
| | CANTX7ID3-0 | CAN Transmit 7 Identifier Register 3~0 | 0xF2 ~ 0xF5 |
| | CANTX7DATA1-8 | CAN Transmit 7 Data Register 1~8 | 0xF6 ~ 0xFD |

**CANTnCFR (0x80,0x90,0xA0,0xB0,0xC0,0xD0,0xE0,0xF0) CAN Transmit n Configuration Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | - | - | - | - | ADTSL | ADMOD | ADEN |
| WR | - | - | - | - | - | ADTSL | ADMOD | ADEN |

ADTSL Auto-dispatch Timer Select
ADTSL=1 Auto dispatch timer 1 is selected.
ADTSL=0 Auto dispatch timer 0 is selected.

ADMOD Auto-dispatch Mode Select
ADMOD=1 indicates the transmit buffer will re-transmit the failed auto-dispatch frame even if a bus error happened.
ADMOD=0 indicates the auto-dispatch frame is a single shot transmit and does not re-transmit the failed frame even if a bus error happened.

ADEN Auto-dispatch Enable
ADEN=1 active transmit buffer as an auto-dispatch transmitter. If auto-dispatch function is active, the CAN controller will periodically transmit the data stored in the transmit buffer to CAN bus.
ADEN=0 indicates the transmit buffer is under normal operation and only the Transmit Request (TR0) command can issue a transmission.

In standard Frame Format (SFF) the identifier consists of 11 bits (ID.28 to ID.18) and in Extended Frame Format (EFF) messages the identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit, which is transmitted first on the bus during the arbitration process. The identifier acts as the message's name and is used for bus arbitration and used in a receiver for acceptance filtering. The smaller the binary value of the identifier is, the higher the priority is. This is due to the larger number of leading dominant bits during arbitration. Also please note the R0, R1 and SRR bits in the SFF and EFF are transmitted as R0=R1=0 and SRR=1.

The transmit buffer layout is subdivided into descriptor and data fields where the first byte of the descriptor field is the frame information byte (frame information). It describes the frame format (SFF or EFF), remote or data frame and the data length. Four identifier bytes for both SFF and EFF messages follow. But only the lower 11 bits in the four

identifier bytes are valid for SFF and the upper 21 bits are invalid. For EFF, the lower 29 bits in the four identifier bytes are valid and the upper 3 bits are invalid. The data field contains up to eight data bytes.

**CANTnFIR (0x81,0x91,0xA1,0xB1,0xC1,0xD1,0xE1,0xF1) CAN Transmit n Frame Information Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | FF | RTR | - | - | DLC[3] | DLC[2] | DLC[1] | DLC[0] |
| WR | FF | RTR | - | - | DLC[3] | DLC[2] | DLC[1] | DLC[0] |

FF — Frame Format
FF=1 indicates Extended format frame
FF=0 indicates Standard format frame

RTR — Remote Frame Transmit Request
RTR=1 indicates remote transmission request frame
RTR=0 indicates regular data frame

DLC[3-0] — Data Byte Count = DLC[3-0]
The number of bytes in the data field of a message is coded by the data length code. At the start of a remote frame transmission the data length code is not considered due to the RTR bit being logic 1 (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the data length code must be specified correctly to avoid bus errors, if two CAN controllers start a remote frame transmission with the same identifier simultaneously. For reasons of compatibility, no data length code > 8 should be used. If a value greater than 8 is used, only 8 bytes of data are transmitted in the data frame with the Data Length Code specified in DLC. For remote frame, DLC[3-0] is used for indicating the requested data length.

In standard Frame Format (SFF) the identifier consists of 11 bits (ID.28 to ID.18) and in Extended Frame Format (EFF) messages the identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit, which is transmitted first on the bus during the arbitration process. The identifier acts as the message's name and is used for bus arbitration and used in a receiver for acceptance filtering. The smaller the binary value of the identifier is, the higher the priority is. This is due to the larger number of leading dominant bits during arbitration. Also please note the R0, R1 and SRR bits in the SFF and EFF are transmitted as R0=R1=0 and SRR=1.

**CANTXnID3 (0x82,0x92,0xA2,0xB2,0xC2,0xD2,0xE2,0xF2) CAN Transmit n Identifier Register 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnID3[7-0] | | | | | | | |
| WR | CANTXnID3[7-0] | | | | | | | |

For SFF ID, CANTXnID3[7-0] is not used.
For EFF ID, CANTXnID3[4-0] = ID[28-24]. CANTXnID3[7-5] is don't care.

**CANTXnID2 (0x83,0x93,0xA3,0xB3,0xC3,0xD3,0xE3,0xF3) CAN Transmit n Identifier Register 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTX0ID2[7-0] | | | | | | | |
| WR | CANTX0ID2[7-0] | | | | | | | |

For SFF ID, CANTXnID2[7-0] is not used.
For EFF ID, CANTXnID2[7-0] = ID[23-16].

**CANTXnID1 (0x84,0x94,0xA4,0xB4,0xC4,0xD4,0xE4,0xF4) CAN Transmit n Identifier Register 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnID1[7-0] | | | | | | | |
| WR | CANTXnID1[7-0] | | | | | | | |

For SFF ID, CANTXnID1[2-0] = ID[28-26]. CANTXnID1[7-3] is not used.
For EFF ID, CANTXnID1[7-0] = ID[15-8].

**CANTXnID0 (0x85,0x95,0xA5,0xB5,0xC5,0xD5,0xE5,0xF5) CAN Transmit n Identifier Register 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnID0[7-0] | | | | | | | |
| WR | CANTXnID0[7-0] | | | | | | | |

For SFF ID, CANTXnID0[7-0] = ID[25-18]
For EFF ID, CANTXnID0[7-0] = ID[7-0].

The data field registers of transmit buffer is mapping to the address following CANTXnID0. The number of transferred data bytes is defined by the data length code. The first bit transmitted is the most significant bit of data byte 1 at the address of “CANTXnID + 1”.

**CANTXnDATA1 (0x86,0x96,0xA6,0xB6,0xC6,0xD6,0xE6,0xF6) CAN Transmit n DATA Register 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA1[7-0] | | | | | | | |
| WR | CANTXnDATA1[7-0] | | | | | | | |

**CANTXnDATA2 (0x87,0x97,0xA7,0xB7,0xC7,0xD7,0xE7,0xF7) CAN Transmit n DATA Register 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA2[7-0] | | | | | | | |
| WR | CANTXnDATA2[7-0] | | | | | | | |

**CANTXnDATA3 (0x88,0x98,0xA8,0xB8,0xC8,0xD8,0xE8,0xF8) CAN Transmit n DATA Register 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA3[7-0] | | | | | | | |
| WR | CANTXnDATA3[7-0] | | | | | | | |

**CANTXnDATA4 (0x89,0x99,0xA9,0xB9,0xC9,0xD9,0xE9,0xF9) CAN Transmit n DATA Register 4 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA4[7-0] | | | | | | | |
| WR | CANTXnDATA4[7-0] | | | | | | | |

**CANTXnDATA5 (0x8A,0x9A,0xAA,0xBA,0xCA,0xDA,0xEA,0xFA) CAN Transmit n DATA Register 5 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA5[7-0] | | | | | | | |
| WR | CANTXnDATA5[7-0] | | | | | | | |

**CANTXnDATA6 (0x8B,0x9B,0xAB,0xBB,0xCB,0xDB,0xEB,0xFB) CAN Transmit n DATA Register 6 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA6[7-0] | | | | | | | |
| WR | CANTXnDATA6[7-0] | | | | | | | |

**CANTXnDATA7 (0x8C,0x9C,0xAC,0xBC,0xCC,0xDC,0xEC,0xFC) CAN Transmit n DATA Register 7 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA7[7-0] | | | | | | | |
| WR | CANTXnDATA7[7-0] | | | | | | | |

**CANTXnDATA8 (0x8D,0x9D,0xAD,0xBD,0xCD,0xDD,0xED,0xFD) CAN Transmit n DATA Register 8 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANTXnDATA8[7-0] | | | | | | | |
| WR | CANTXnDATA8[7-0] | | | | | | | |

The dispatch function allows the CAN controller to send out periodic dispatch frame without software interactions. Each transmit buffer can be configured as a dispatcher by setting ADEN. The layout of this dispatch transmit buffer is the same as the transmit buffer consisting of a transmit frame information register, four transmit ID registers, and 8 transmit data registers. The interval of the dispatch can be defined by either dispatch timers, CAND0TR, or by CAND1TR. All these register can be written only in reset mode. The Transmitter 0 (Dispatcher 0) has the highest transmit priority, and the Transmitter 1 (Dispatcher 1) has the second transmit priority, etc. And the Transmitter 7 (Dispatcher 7) has the lowest transmit priority.

**CAND0TR0 (0x0E) CAN Dispatch Interval Timer 0 Low Byte Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CAND0TR[7:0] | | | | | | | |
| WR | CAND0TR[7:0] | | | | | | | |

This register can be modified only in reset mode.
CAND0TR[7-0] Dispatch Timer 0 bit 7 to 0

**CAND0TR1 (0x0F) CAN Dispatch Interval Timer 0 High Byte Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CAND0TR[15-8] | | | | | | | |
| WR | CAND0TR[15-8] | | | | | | | |

This register can be modified only in reset mode.
CAND0TR[15-8] Dispatch Timer 0 bit 15 to 8

The interval defined by CAND0TR is "(CAND0TR[15-0] + 16) * 100 * BT", where BT is the bit time of CAN bus. Because the length of Dispatch Interval Timer 0 Register is 16, the maximal programmable time period of the dispatch timer is ($2^{16}$+16) * 100 * (1 / 1Mbps) = 6.55sec under 1Mbps bit rate.
Dispatch interval timer 0 is disabled when CAND0TR[15-0] = 0x0000.

**CAND1TR0 (0x1E) CAN Dispatch Interval Timer 1 Low Byte Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CAND1TR[7:0] | | | | | | | |
| WR | CAND1TR[7:0] | | | | | | | |

This register can be modified only in reset mode.
CAND1TR[7-0] Dispatch Timer 1 bit 7 to 0

**CAND1TR1 (0x1F) CAN Dispatch Interval Timer 1 High Byte Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CAND1TR[15-8] | | | | | | | |
| WR | CAND1TR[15-8] | | | | | | | |

This register can be modified only in reset mode.
CAND0TR[15-8] Dispatch Timer 1 bit 15 to 8

The interval defined by CAND1TR is "(CAND1TR[15-0] + 16) * 100 * BT", where BT is the bit time of CAN bus. Because the length of Dispatch Interval Timer 1 Register is 16, the maximal programmable time period of the dispatch timer is ($2^{16}$+16) * 100 * (1 / 1Mbps) = 6.55sec under 1Mbps bit rate.
Dispatch interval timer 1 is disabled when CAND1TR[15-0] = 0x0000.

**CANTCR0/CANTSR0 (0x05) CAN Transmit Request/Status 0 Register RW (0x0F)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | TBB3 | TBB2 | TBB1 | TBB0 | TCS3 | TCS2 | TCS1 | TCS0 |
| WR | TAB3 | TAB2 | TAB1 | TAB0 | TR3 | TR2 | TR1 | TR0 |

The CAN Transmit Request 0 register (CANTCR0) and the CAN Transmit Status 0 register (CANTSR0) share the address 0x05. The CANTCR0 is a write-only register and the CANTSR0 is a read-only register.

TBB3 Transmit Buffer 3 Busy Status (read-only)
TBB3=1 indicates a message in the transmit buffer 3 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR3=1 or dispatch transmission is in progress.
TBB3=0 indicates the transmit buffer 3 is valid to be programmed a new message by MCU.

TBB2 Transmit Buffer 2 Busy Status (read-only)
TBB2=1 indicates a message in the transmit buffer 2 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR2=1 or dispatch transmission is in progress.
TBB2=0 indicates the transmit buffer 2 is valid to be programmed a new message by MCU.

TBB1 Transmit Buffer 1 Busy Status (read-only)
TBB1=1 indicates a message in the transmit buffer 1 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR1=1 or dispatch transmission is in progress.
TBB1=0 indicates the transmit buffer 1 is valid to be programmed a new message by MCU.

TBB0 Transmit Buffer 0 Busy Status (read-only)
TBB0=1 indicates a message in the transmit buffer 0 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR0=1 or dispatch transmission is in progress.
TBB0=0 indicates the transmit buffer 0 is valid to be programmed a new message by MCU.

TCS3 Transmit Complete of transmit buffer 3 (read-only)
TCS3=1 when the last requested or dispatch transmission has been successfully completed or aborted.
TCS3=0 indicates the transmission of transmit buffer 3 is in progress.

TCS2 Transmit Complete of transmit buffer 2 (read-only)
TCS2=1 when the last requested or dispatch transmission has been successfully completed or aborted.
TCS2=0 indicates the transmission of transmit buffer 2 is in progress.

TCS1 Transmit Complete of transmit buffer 1 (read-only)
TCS1=1 when the last requested or dispatch transmission has been successfully completed or aborted.
TCS1=0 indicates the transmission of transmit buffer 1 is in progress.

TCS0 Transmit Complete of transmit buffer 0 (read-only)
TCS0=1 when the last requested or dispatch transmission has been successfully completed or aborted.
TCS0=0 indicates the transmission of transmit buffer 0 is in progress.

TAB3 Abort Transmitting Command of transmit buffer 3 (write-only)
TAB3=1 will abort the pending transmission request issued from transmit buffer 3. TAB3 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB3 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB3 is self-cleared by hardware.

TAB2 Abort Transmitting Command of transmit buffer 2 (write-only)
TAB2=1 will abort the pending transmission request issued from transmit buffer 2. TAB2 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB2 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB2 is self-cleared by hardware.

TAB1 Abort Transmitting Command of transmit buffer 1 (write-only)
TAB1=1 will abort the pending transmission request issued from transmit buffer 1. TAB1 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB1 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB1 is self-cleared by hardware.

TAB0 Abort Transmitting Command of transmit buffer 0 (write-only)

TAB0=1 will abort the pending transmission request issued from transmit buffer 0. TAB0 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB0 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB0 is self-cleared by hardware.

TR3 Transmit Request Command of transmit buffer 3 (write-only)
TR3=1 to request the CAN controller transmits the message stored in the transmit buffer 3. A transmitting message only can be cancelled by setting TAB3. User can check TBB3 and TCS3 to know this transmission is in progress or completed.

TR2 Transmit Request Command of transmit buffer 2 (write-only)
TR2=1 to request the CAN controller transmits the message stored in the transmit buffer 2. A transmitting message only can be cancelled by setting TAB2. User can check TBB2 and TCS2 to know this transmission is in progress or completed.

TR1 Transmit Request Command of transmit buffer 1 (write-only)
TR1=1 to request the CAN controller transmits the message stored in the transmit buffer 1. A transmitting message only can be cancelled by setting TAB1. User can check TBB1 and TCS1 to know this transmission is in progress or completed.

TR0 Transmit Request Command of transmit buffer 0 (write-only)
TR0=1 to request the CAN controller transmits the message stored in the transmit buffer 0. A transmitting message only can be cancelled by setting TAB0. User can check TBB0 and TCS0 to know this transmission is in progress or completed.

**CANTCR1/CANTSR1 (0x06) CAN Transmit Request/Status 1 Register RW (0x0F)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | TBB7 | TBB6 | TBB5 | TBB4 | TCS7 | TCS6 | TCS5 | TCS4 |
| WR | TAB7 | TAB6 | TAB5 | TAB4 | TR7 | TR6 | TR5 | TR4 |

The CAN Transmit Request 1 register (CANTCR1) and the CAN Transmit Status 1 register (CANTSR1) share the address 0x06. The CANTCR1 is a write-only register and the CANTSR1 is a read-only register.

TBB7 Transmit Buffer 7 Busy Status (read-only)
TBB7=1 indicates a message in the transmit buffer 7 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR7=1 or dispatch transmission is in progress.
TBB7=0 indicates the transmit buffer 7 is valid to be programmed a new message by MCU.

TBB6 Transmit Buffer 6 Busy Status (read-only)
TBB6=1 indicates a message in the transmit buffer 6 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR6=1 or dispatch transmission is in progress.
TBB6=0 indicates the transmit buffer 6 is valid to be programmed a new message by MCU.

TBB5 Transmit Buffer 5 Busy Status (read-only)
TBB5=1 indicates a message in the transmit buffer 5 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR5=1 or dispatch transmission is in progress.
TBB5=0 indicates the transmit buffer 5 is valid to be programmed a new message by MCU.

TBB4 Transmit Buffer 4 Busy Status (read-only)
TBB4=1 indicates a message in the transmit buffer 4 is either waiting for transmission or in progress of being transmitted and the MCU cannot access that transmit buffer. This bit will be set when TR4=1 or dispatch transmission is in progress.
TBB4=0 indicates the transmit buffer 4 is valid to be programmed a new message by MCU.

TCS7 Transmit Complete of transmit buffer 7 (read-only)
TCS7=1 when the last requested or dispatch transmission has been successfully completed or aborted.
TCS7=0 indicates the transmission of transmit buffer 7 is in progress.

TCS6 Transmit Complete of transmit buffer 6 (read-only)
TCS6=1 when the last requested or dispatch transmission has been successfully completed or aborted.
TCS6=0 indicates the transmission of transmit buffer 6 is in progress.

TCS5 Transmit Complete of transmit buffer 5 (read-only)

| | |
|---|---|
| | TCS5=1 when the last requested or dispatch transmission has been successfully completed or aborted.<br>TCS5=0 indicates the transmission of transmit buffer 5 is in progress. |
| TCS4 | Transmit Complete of transmit buffer 4 (read-only)<br>TCS4=1 when the last requested or dispatch transmission has been successfully completed or aborted.<br>TCS4=0 indicates the transmission of transmit buffer 4 is in progress. |
| TAB7 | Abort Transmitting Command of transmit buffer 7 (write-only)<br>TAB7=1 will abort the pending transmission request issued from transmit buffer 7. TAB7 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB7 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB7 is self-cleared by hardware. |
| TAB6 | Abort Transmitting Command of transmit buffer 6 (write-only)<br>TAB6=1 will abort the pending transmission request issued from transmit buffer 6. TAB6 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB6 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB6 is self-cleared by hardware. |
| TAB5 | Abort Transmitting Command of transmit buffer 5 (write-only)<br>TAB5=1 will abort the pending transmission request issued from transmit buffer 5. TAB5 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB5 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB5 is self-cleared by hardware. |
| TAB4 | Abort Transmitting Command of transmit buffer 4 (write-only)<br>TAB4=1 will abort the pending transmission request issued from transmit buffer 4. TAB4 does not stop an on-going message transmission, but the next transmission attempt will be aborted. TAB4 should be used only for aborting a message transmission with errors (mostly in ACK error). TAB4 is self-cleared by hardware. |
| TR7 | Transmit Request Command of transmit buffer 7 (write-only)<br>TR7=1 to request the CAN controller transmits the message stored in the transmit buffer 7. A transmitting message only can be cancelled by setting TAB7. User can check TBB7 and TCS7 to know this transmission is in progress or completed. |
| TR6 | Transmit Request Command of transmit buffer 6 (write-only)<br>TR6=1 to request the CAN controller transmits the message stored in the transmit buffer 6. A transmitting message only can be cancelled by setting TAB6. User can check TBB6 and TCS6 to know this transmission is in progress or completed. |
| TR5 | Transmit Request Command of transmit buffer 5 (write-only)<br>TR5=1 to request the CAN controller transmits the message stored in the transmit buffer 5. A transmitting message only can be cancelled by setting TAB5. User can check TBB5 and TCS5 to know this transmission is in progress or completed. |
| TR4 | Transmit Request Command of transmit buffer 4 (write-only)<br>TR4=1 to request the CAN controller transmits the message stored in the transmit buffer 4. A transmitting message only can be cancelled by setting TAB4. User can check TBB4 and TCS4 to know this transmission is in progress or completed. |

## 1.6 Receive FIFO

In IS31IO8972, there is a 1024-byte data FIFO for receive frame. The maximal byte number of a receive frame is 14, including one Receive Frame Filter Number (RFFNR), one Receive Frame Information Register (RFIFR), four Receive Frame Identifier Registers (RFIDR), and maximal eight Receive Frame Data Registers (RFDTR). The minimum valid receive frame in the Receive FIFO is 73 in the same time without overload error. MCU can read the earliest received frame from the Receive Frame Buffer located from address 0x10 to 0x1D. After reading the contents of the receive buffer, MCU must release this memory space in the RX FIFO by setting the Release Receive Buffer bit in Command Register. After releasing the current receive buffer, the data of next frame will be popped to address 0x10 to 0x1D. All registers in Receive Frame Buffer are read only.
The following diagram shows the frame arrangement.

| LOCATION | DESCRIPTION | BIT [7-0] | | | | |
|---|---|---|---|---|---|---|
| 0x10 | ACP FILTER | - | - | - | - | ACPFIL[3-0] |
| 0x11 | RX INFO | IDE | RTR | 0 | 0 | DLC[3-0] |
| 0x12 | ID3 | 0 | 0 | 0 | ID[28-24] | |
| 0x13 | ID2 | ID[23-16] | | | | |
| 0x14 | ID1 | ID[15-8] | | | | |
| 0x15 | ID0 | ID[7-0] | | | | |
| 0x16 | DATA1 | DATA1[7-0] | | | | |
| 0x17 | DATA2 | DATA2[7-0] | | | | |
| 0x18 | DATA3 | DATA3[7-0] | | | | |
| 0x19 | DATA4 | DATA4[7-0] | | | | |
| 0x1A | DATA5 | DATA5[7-0] | | | | |
| 0x1B | DATA6 | DATA6[7-0] | | | | |
| 0x1C | DATA7 | DATA7[7-0] | | | | |
| 0x1D | DATA8 | DATA8[7-0] | | | | |

**CANACPFIL (0x10) CAN Receive Acceptance Filter Number Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | 0 | 1 | 0 | 0 | ACPFIL[3] | ACPFIL[2] | ACPFIL[1] | ACPFIL[0] |
| WR | - | - | - | - | - | - | - | - |

ACPFIL[3-0] CAN Receive Acceptance Filter Number
ACPFIL[3-0] contains the Acceptance Filter number that matches with the received frame.
CANACPFIL[7-4] are reserved bits and the constant “0100” always be read.

**CANRXINFO (0x11) CAN Receive Information Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | IDE | RTR | 0 | 0 | DLC[3] | DLC[2] | DLC[1] | DLC[0] |
| WR | - | - | - | - | - | - | - | - |

CANRXINFO contains the information of the received frame, includes IDE, RTR, and DLC[3-0]. IDE bit determines whether the frame is SFF (IDE=0) or EFF (IDE=1). For SFF, RXID[28-11] will be all 0 and only RXID[10-0] is meaningful. The received data bytes depend on DLC code. For unspecified data bytes and reserved bytes, the contents are not guaranteed. Please note the R0, R1 and SRR bits along with CRC field are not stored.

IDE Frame Format
IDE=1: Extended format frame with 29-bit ID valid
IDE=0: Standard format frame with 11-bit ID valid

RTR Remote Frame Transmission
RTR=1: A remote frame transmission was received and the data length code (DLC) is not considered. This forces the number of transmitted/received data bytes to be 0.
RTR=0: A data frame was received and the data length code (DLC) must be specified correctly.

DLC[3-0] Data Length Code
Specify the byte number of the received data frame.
DataByteCount = 8 X DLC[3] + 4 X DLC[2] + 2 X DLC[1] + DLC[0]
For reasons of compatibility, no DLC > 8 should be used. If a value >8 is selected, only 8 bytes are transmitted and received in the data frame.

**CANRXID3 (0x12) CAN Receive ID3 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | 0 | 0 | 0 | RXID[28] | RXID[27] | RXID[26] | RXID[25] | RXID[24] |
| WR | - | - | - | - | - | - | - | - |

For SFF ID, CANRXID3[7-0] is not used.
For EFF ID, CANRXID3[4-0] = RXID[28-24]. CANRXID3[7-5] is don’t care.

**CANRXID2 (0x13) CAN Receive ID2 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | RXID[23] | RXID[22] | RXID[21] | RXID[20] | RXID[19] | RXID[18] | RXID[17] | RXID[16] |
| WR | - | - | - | - | - | - | - | - |

For SFF ID, CANRXID2[7-0] is not used.
For EFF ID, CANRXID2[7-0] = RXID[23-16].

**CANRXID1 (0x14) CAN Receive ID1 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | RXID[15] | RXID[14] | RXID[13] | RXID[12] | RXID[11] | RXID[10] | RXID[9] | RXID[8] |
| WR | - | - | - | - | - | - | - | - |

For SFF ID, CANRXID1[2-0] = RXID[10-8]. CANRXID1[7-3] is don't care.
For EFF ID, CANRXID1[7-0] = RXID[15-8].

**CANRXID0 (0x15) CAN Receive ID0 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | RXID[7] | RXID[6] | RXID[5] | RXID[4] | RXID[3] | RXID[2] | RXID[1] | RXID[0] |
| WR | - | - | - | - | - | - | - | - |

For SFF ID, CANRXID0[7-0] = RXID[7-0].
For EFF ID, CANRXID0[7-0] = RXID[7-0].

Data field registers of the receive buffer is mapping to address 16H to 1DH. The number of transferred data bytes is defined by the data length code (DLC[3-0]) an only the number of bytes stored in the Receive Frame Data Field Register is valid. The first bit received is the most significant bit of data byte 0 stored at address 16H.

**CANRXDTA0 (0x16) CAN Receive Data 0 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA0[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA1 (0x17) CAN Receive Data 1 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA1[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA2 (0x18) CAN Receive Data 2 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA2[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA3 (0x19) CAN Receive Data 3 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA3[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA4 (0x1A) CAN Receive Data 4 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA4[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA5 (0x1B) CAN Receive Data 5 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA5[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA6 (0x1C) CAN Receive Data 6 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA6[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXDTA7 (0x1D) CAN Receive Data 7 Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXDATA7[7-0] | | | | | | | |
| WR | - | | | | | | | |

**CANRXFCNT (0x7A) CAN Receive Frame Count Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANRXFCNT[7-0] | | | | | | | |
| WR | - | | | | | | | |

CANRXFCNT CANRXFCNT[7-0] indicates the number of all unread received frame in the receive buffer. This register will be increased by one when receive a valid frame and will be decreased by one when setting the Release Receive Buffer (RRB) bit to logic 1. When the register reaches zero means the receive buffer is empty.

## 1.7 Acceptance Filter

There are 12 Acceptance Filters and 8 Filter Masks for matching with the ID of a message on the bus. Each filter can be enabled or disabled independently. Each acceptance filter is consisted of 4 ID registers and corresponding ID Mask register to mask the match comparing the specific bit in the ID. The layout of the ID registers corresponding to forming ID28 to ID1 for SFF and EFF is the same as Transmit ID registers. Please note the ID registers only defines the ID and do not define the frame type, therefore it is possible a value in the ID registers corresponding to two possible matches for SFF and EFF separately. This is especially true since the ID bit ordering for SFF and EFF is opposite, the user should check if unintended accepted ID is possible for another frame type. For the Filter's Mask register, when mask bit is 0 the corresponding bit is compared for matching, and when the mask bit is 1 the corresponding bit is ignored. All Acceptance Filter ID registers and Mask register can only be modified in the reset mode, and appear as read-only in normal operating mode.

**CANFTERL (0x70) CAN ID Filter Low Enable Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | FTER7 | FTER6 | FTER5 | FTER4 | FTER3 | FTER2 | FTER1 | FTER0 |
| WR | FTER7 | FTER6 | FTER5 | FTER4 | FTER3 | FTER2 | FTER1 | FTER0 |

There are 12 Acceptance Filters for message ID matching. This register controls the enabling of each filter, multiple filters can be enabled. This register can be modified only in reset mode and appears as read-only in normal operating mode.

FTER7 Acceptance Filter 7 Enable
FTER7=1 enables Acceptance Filter 7

FTER6 Acceptance Filter 6 Enable
FTER6=1 enables Acceptance Filter 6

FTER5 Acceptance Filter 5 Enable
FTER1=5 enables Acceptance Filter 5

FTER4 Acceptance Filter 4 Enable
FTER4=1 enables Acceptance Filter 4

FTER3 Acceptance Filter 3 Enable
FTER3=1 enables Acceptance Filter 3

| | |
|---|---|
| FTER2 | Acceptance Filter 2 Enable<br>FTER2=1 enables Acceptance Filter 2 |
| FTER1 | Acceptance Filter 1 Enable<br>FTER1=1 enables Acceptance Filter 1 |
| FTER0 | Acceptance Filter 0 Enable<br>FTER0=1 enables Acceptance Filter 0 |

**CANFTERH (0x71) CAN ID Filter High Enable Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | - | - | - | FTER11 | FTER10 | FTER9 | FTER8 |
| WR | - | - | - | - | FTER11 | FTER10 | FTER9 | FTER8 |

| | |
|---|---|
| FTER11 | Acceptance Filter 11 Enable<br>FTER11=1 enables Acceptance Filter 11 |
| FTER10 | Acceptance Filter 10 Enable<br>FTER10=1 enables Acceptance Filter 10 |
| FTER9 | Acceptance Filter 9 Enable<br>FTER9=1 enables Acceptance Filter 9 |
| FTER8 | Acceptance Filter 8 Enable<br>FTER8=1 enables Acceptance Filter 8 |

In IS31IO8972, there are 12 ID Acceptance Filters and 8 Filter Masks, but not all Receive Filter Acceptances can be enabled at the same time. The minimal time period between a valid ID and a CRC delimiter is 20-bit time and Filter Controller must compare all enabled Filters during the period. So the maximal number of available Receive Filter is limited by the frequency of the system clock, CAN's bit rate and data format. Following table lists the maximal number of available Receive Filter with different operation conditions.

| | Frequency of System Clock / Frequency of CAN bit rate | | | |
|---|---|---|---|---|
| | 8 | 10 | 12 | 16 |
| Standard Format (11-bit ID) | 11 | 12 | 12 | 12 |
| Extended Format (29-bit ID) | 7 | 9 | 11 | 12 |

**CANRFIE (0x72) CAN Acceptance Filter Match Interrupt Enable Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | RFIE7 | RFIE6 | RFIE5 | RFIE4 | RFIE3 | RFIE2 | RFIE1 | RFIE0 |
| WR | RFIE7 | RFIE6 | RFIE5 | RFIE4 | RFIE3 | RFIE2 | RFIE1 | RFIE0 |

| | |
|---|---|
| RFIE7 | RFIE7=1 Enable Acceptance Filter 7 match interrupt<br>RFIE7=0 Disable Acceptance Filter 7 match interrupt |
| RFIE6 | RFIE6=1 Enable Acceptance Filter 6 match interrupt<br>RFIE6=0 Disable Acceptance Filter 6 match interrupt |
| RFIE5 | RFIE5=1 Enable Acceptance Filter 5 match interrupt<br>RFIE5=0 Disable Acceptance Filter 5 match interrupt |
| RFIE4 | RFIE4=1 Enable Acceptance Filter 4 match interrupt<br>RFIE4=0 Disable Acceptance Filter 4 match interrupt |
| RFIE3 | RFIE3=1 Enable Acceptance Filter 3 match interrupt<br>RFIE3=0 Disable Acceptance Filter 3 match interrupt |
| RFIE2 | RFIE2=1 Enable Acceptance Filter 2 match interrupt<br>RFIE2=0 Disable Acceptance Filter 2 match interrupt |
| RFIE1 | RFIE1=1 Enable Acceptance Filter 1 match interrupt<br>RFIE1=0 Disable Acceptance Filter 1 match interrupt |
| RFIE0 | RFIE0=1 Enable Acceptance Filter 0 match interrupt<br>RFIE0=0 Disable Acceptance Filter 0 match interrupt |

**CANEXIE (0x73) CAN Extra Interrupt Enable Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | WKUPIE | PHYIE | ECCIE | THMIE | RFIE11 | RFIE10 | RFIE9 | RFIE8 |
| WR | WKUPIE | PHYIE | ECCIE | THMIE | RFIE11 | RFIE10 | RFIE9 | RFIE8 |

| | |
|---|---|
| WKUPIE | WKUPIE=1 Enable external wakeup interrupt. When detect an external wakeup signal, IS31IO8972 wakeup from stop mode and notice CPU by pulling low the INTn pin.<br>WKUPIE=0 Disable external wakeup interrupt |
| PHYIE | PHYIE=1 Enable transceiver error interrupt. When detect an error state on the transceiver, IS31IO8972 will notice CPU by pulling low the INTn pin. |
| ECCIE | ECCIE=1 Enable SRAM ECC correction interrupt. When the content of SRAM was corrected by built-in Hamming code, IS31IO8972 will notice CPU by pulling low the INTn pin. |
| THMIE | THMIE=1 Enable the transceiver's built-in thermal reset interrupt. When the transceiver is shut-down because of the thermal reset, IS31IO8972 will notice CPU by pulling low the INTn pin. |
| RFIE11 | RFIE11=1 Enable Acceptance Filter 11 match interrupt<br>RFIE11=0 Disable Acceptance Filter 11 match interrupt |
| RFIE10 | RFIE10=1 Enable Acceptance Filter 10 match interrupt<br>RFIE10=0 Disable Acceptance Filter 10 match interrupt |
| RFIE9 | RFIE9=1 Enable Acceptance Filter 9 match interrupt<br>RFIE9=0 Disable Acceptance Filter 9 match interrupt |
| RFIE8 | RFIE8=1 Enable Acceptance Filter 8 match interrupt<br>RFIE8=0 Disable Acceptance Filter 8 match interrupt |

**CANRFIF (0x74) CAN Acceptance Filter Match Interrupt Flag Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | RFIF7 | RFIF6 | RFIF5 | RFIF4 | RFIF3 | RFIF2 | RFIF1 | RFIF0 |
| WR | - | - | - | - | - | - | - | - |

This is a read-only register. All flags are event change triggering while the received ID match the Acceptance Filter. All bits are cleared after read.

| | |
|---|---|
| RFIF7 | RFIF7 is set when FTER7=1 and the ID of received frame matches Acceptance Filter 7.<br>RFIF7 is cleared after read. |
| RFIF6 | RFIF6 is set when FTER6=1 and the ID of received frame matches Acceptance Filter 6.<br>RFIF6 is cleared after read. |
| RFIF5 | RFIF5 is set when FTER5=1 and the ID of received frame matches Acceptance Filter 5.<br>RFIF5 is cleared after read. |
| RFIF4 | RFIF4 is set when FTER4=1 and the ID of received frame matches Acceptance Filter 4.<br>RFIF4 is cleared after read. |
| RFIF3 | RFIF3 is set when FTER3=1 and the ID of received frame matches Acceptance Filter 3.<br>RFIF3 is cleared after read. |
| RFIF2 | RFIF2 is set when FTER2=1 and the ID of received frame matches Acceptance Filter 2.<br>RFIF2 is cleared after read. |
| RFIF1 | RFIF1 is set when FTER1=1 and the ID of received frame matches Acceptance Filter 1.<br>RFIF1 is cleared after read. |
| RFIF0 | RFIF0 is set when FTER0=1 and the ID of received frame matches Acceptance Filter 0.<br>RFIF0 is cleared after read. |

**CANEXIF (0x75) CAN Extra Interrupt Flag Register RO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | WKUPIF | PHYIF | ECCIF | THMIF | RFIF11 | RFIF10 | RFIF9 | RFIF8 |
| WR | - | - | - | - | - | - | - | - |

WKUPIF External Wakeup Interrupt Flag
WKUPIF is set when detect an external wakeup signal; IS31IO8972 will wake up from stop mode and notice CPU by pulling low the INTn pin if WKUPIE was set. WKUPIF is cleared after read.

PHYIF Transceiver Error Interrupt Flag
PHYIF is set when detect an error state on the transceiver; IS31IO8972 will notice CPU by pulling low the INTn pin if PHYIE was set. The fault status can be read from the TRANFAULT register. PHYIF is cleared only when all faults have been banished from CAN bus and all bits in TRANFAULT are cleared by software.

ECCIF ECCIF is set when the content of SRAM was corrected by built-in Hamming code; IS31IO8972 will notice CPU by pulling low the INTn pin if ECCIE was set. ECCIF is cleared after read.

THMIF THMIF is set when the transceiver is shut-down due to the built-in thermal reset happened or recovered from thermal shut-down state. IS31IO8972 will notice CPU by pulling low the INTn pin if THMIE was set. THMIF is cleared after read.

RFIF11 RFIF11 is set when FTER11=1 and the ID of received frame matches Acceptance Filter 11. RFIF11 is cleared after read.

RFIF10 RFIF10 is set when FTER10=1 and the ID of received frame matches Acceptance Filter 10. RFIF10 is cleared after read.

RFIF9 RFIF9 is set when FTER9=1 and the ID of received frame matches Acceptance Filter 9. RFIF9 is cleared after read.

RFIF8 RFIF8 is set when FTER8=1 and the ID of received frame matches Acceptance Filter 8. RFIF8 is cleared after read.

All Acceptance Filter registers can be modified only in reset mode and appears as read-only in normal operating mode.

**CANAFID03 (0x20) Acceptance Filter 0 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL0[2-0] | | | CANFID03[4-0] | | | | |
| WR | CANMKSEL0[2-0] | | | CANFID03[4-0] | | | | |

CANMKSEL0 Acceptance Filter 0's Mask Selection
Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Mask by CANMKSEL0[2-0].
CANMKSEL0[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL0[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL0[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL0[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL0[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL0[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL0[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL0[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID02 (0x21) Acceptance Filter 0 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID02[7-0] | | | | | | | |
| WR | CANFID02[7-0] | | | | | | | |

**CANAFID01 (0x22) Acceptance Filter 0 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID01[7-0] | | | | | | | |
| WR | CANFID01[7-0] | | | | | | | |

**CANAFID00 (0x23) Acceptance Filter 0 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID00[7-0] | | | | | | | |
| WR | CANFID00[7-0] | | | | | | | |

**CANAFID13 (0x24) Acceptance Filter 1 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL1[2-0] | | | CANFID13[4-0] | | | | |
| WR | CANMKSEL1[2-0] | | | CANFID13[4-0] | | | | |

CANMKSEL1 Acceptance Filter 1's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL1[2-0].

CANMKSEL1[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL1[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL1[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL1[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL1[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL1[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL1[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL1[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID12 (0x25) Acceptance Filter 1 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID12[7-0] | | | | | | | |
| WR | CANFID12[7-0] | | | | | | | |

**CANAFID11 (0x26) Acceptance Filter 1 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID11[7-0] | | | | | | | |
| WR | CANFID11[7-0] | | | | | | | |

**CANAFID10 (0x27) Acceptance Filter 1 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID10[7-0] | | | | | | | |
| WR | CANFID10[7-0] | | | | | | | |

**CANAFID23 (0x28) Acceptance Filter 2 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL2[2-0] | | | CANFID23[4-0] | | | | |
| WR | CANMKSEL2[2-0] | | | CANFID23[4-0] | | | | |

CANMKSEL2 Acceptance Filter 2's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL2[2-0].

CANMKSEL2[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL2[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL2[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL2[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL2[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL2[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL2[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL2[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID22 (0x29) Acceptance Filter 2 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID22[7-0] | | | | | | | |
| WR | CANFID22[7-0] | | | | | | | |

**CANAFID21 (0x2A) Acceptance Filter 2 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID21[7-0] | | | | | | | |
| WR | CANFID21[7-0] | | | | | | | |

**CANAFID20 (0x2B) Acceptance Filter 2 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID20[7-0] | | | | | | | |
| WR | CANFID20[7-0] | | | | | | | |

**CANAFID33 (0x2C) Acceptance Filter 3 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL3[2-0] | | | CANFID33[4-0] | | | | |
| WR | CANMKSEL3[2-0] | | | CANFID33[4-0] | | | | |

CANMKSEL3 Acceptance Filter 3's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL3[2-0].

CANMKSEL3[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL3[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL3[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL3[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL3[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL3[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL3[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL3[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID32 (0x2D) Acceptance Filter 3 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID32[7-0] | | | | | | | |
| WR | CANFID32[7-0] | | | | | | | |

**CANAFID31 (0x2E) Acceptance Filter 3 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID31[7-0] | | | | | | | |
| WR | CANFID31[7-0] | | | | | | | |

**CANAFID30 (0x2F) Acceptance Filter 3 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID30[7-0] | | | | | | | |
| WR | CANFID30[7-0] | | | | | | | |

**CANAFID43 (0x30) Acceptance Filter 4 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL4[2-0] | | | CANFID43[4-0] | | | | |
| WR | CANMKSEL4[2-0] | | | CANFID43[4-0] | | | | |

CANMKSEL4 Acceptance Filter 4's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL4[2-0].

CANMKSEL4[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL4[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL4[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL4[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL4[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL4[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL4[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL4[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID42 (0x31) Acceptance Filter 4 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID42[7-0] | | | | | | | |
| WR | CANFID42[7-0] | | | | | | | |

**CANAFID41 (0x32) Acceptance Filter 4 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID41[7-0] | | | | | | | |
| WR | CANFID41[7-0] | | | | | | | |

**CANAFID40 (0x33) Acceptance Filter 4 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID40[7-0] | | | | | | | |
| WR | CANFID40[7-0] | | | | | | | |

**CANAFID53 (0x34) Acceptance Filter 5 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL5[2-0] | | | CANFID53[4-0] | | | | |
| WR | CANMKSEL5[2-0] | | | CANFID53[4-0] | | | | |

CANMKSEL5 Acceptance Filter 5's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL5[2-0].

CANMKSEL5[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL5[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL5[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL5[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL5[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL5[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL5[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL5[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID52 (0x35) Acceptance Filter 5 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID52[7-0] | | | | | | | |
| WR | CANFID52[7-0] | | | | | | | |

**CANAFID51 (0x36) Acceptance Filter 5 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID51[7-0] | | | | | | | |
| WR | CANFID51[7-0] | | | | | | | |

**CANAFID50 (0x37) Acceptance Filter 5 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID50[7-0] | | | | | | | |
| WR | CANFID50[7-0] | | | | | | | |

**CANAFID63 (0x38) Acceptance Filter 6 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL6[2-0] | | | CANFID63[4-0] | | | | |
| WR | CANMKSEL6[2-0] | | | CANFID63[4-0] | | | | |

CANMKSEL6 Acceptance Filter 6's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL6[2-0].

CANMKSEL6[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL6[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL6[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL6[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL6[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL6[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL6[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL6[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID62 (0x39) Acceptance Filter 6 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID62[7-0] | | | | | | | |
| WR | CANFID62[7-0] | | | | | | | |

**CANAFID61 (0x3A) Acceptance Filter 6 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID61[7-0] | | | | | | | |
| WR | CANFID61[7-0] | | | | | | | |

**CANAFID60 (0x3B) Acceptance Filter 6 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID60[7-0] | | | | | | | |
| WR | CANFID60[7-0] | | | | | | | |

**CANAFID73 (0x3C) Acceptance Filter 7 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL7[2-0] | | | CANFID73[4-0] | | | | |
| WR | CANMKSEL7[2-0] | | | CANFID73[4-0] | | | | |

CANMKSEL7 Acceptance Filter 7's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL7[2-0].

CANMKSEL7[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL7[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL7[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL7[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL7[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL7[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL7[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL7[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID72 (0x3D) Acceptance Filter 7 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID72[7-0] | | | | | | | |
| WR | CANFID72[7-0] | | | | | | | |

**CANAFID71 (0x3E) Acceptance Filter 7 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID71[7-0] | | | | | | | |
| WR | CANFID71[7-0] | | | | | | | |

**CANAFID70 (0x3F) Acceptance Filter 7 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID70[7-0] | | | | | | | |
| WR | CANFID70[7-0] | | | | | | | |

**CANAFID83 (0x40) Acceptance Filter 8 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL8[2-0] | | | CANFID83[4-0] | | | | |
| WR | CANMKSEL8[2-0] | | | CANFID83[4-0] | | | | |

CANMKSEL8 Acceptance Filter 8's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL8[2-0].

CANMKSEL8[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL8[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL8[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL8[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL8[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL8[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL8[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL8[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID82 (0x41) Acceptance Filter 8 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID82[7-0] | | | | | | | |
| WR | CANFID82[7-0] | | | | | | | |

**CANAFID81 (0x42) Acceptance Filter 8 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID81[7-0] | | | | | | | |
| WR | CANFID81[7-0] | | | | | | | |

**CANAFID80 (0x43) Acceptance Filter 8 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID80[7-0] | | | | | | | |
| WR | CANFID80[7-0] | | | | | | | |

**CANAFID93 (0x44) Acceptance Filter 9 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL9[2-0] | | | CANFID93[4-0] | | | | |
| WR | CANMKSEL9[2-0] | | | CANFID93[4-0] | | | | |

CANMKSEL9 Acceptance Filter 9's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL9[2-0].

CANMKSEL9[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL9[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL9[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL9[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL9[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL9[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL9[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL9[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID92 (0x45) Acceptance Filter 9 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID92[7-0] | | | | | | | |
| WR | CANFID92[7-0] | | | | | | | |

**CANAFID91 (0x46) Acceptance Filter 9 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID91[7-0] | | | | | | | |
| WR | CANFID91[7-0] | | | | | | | |

**CANAFID90 (0x47) Acceptance Filter 9 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID90[7-0] | | | | | | | |
| WR | CANFID90[7-0] | | | | | | | |

**CANAFID103 (0x48) Acceptance Filter 10 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL10[2-0] | | | CANFID103[4-0] | | | | |
| WR | CANMKSEL10[2-0] | | | CANFID103[4-0] | | | | |

CANMKSEL10 Acceptance Filter 10's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL10[2-0].

CANMKSEL10[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL10[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL10[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL10[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL10[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL10[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL10[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL10[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID102 (0x49) Acceptance Filter 10 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID102[7-0] | | | | | | | |
| WR | CANFID102[7-0] | | | | | | | |

**CANAFID101 (0x4A) Acceptance Filter 10 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID101[7-0] | | | | | | | |
| WR | CANFID101[7-0] | | | | | | | |

**CANAFID100 (0x4B) Acceptance Filter 10 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID100[7-0] | | | | | | | |
| WR | CANFID100[7-0] | | | | | | | |

**CANAFID113 (0x4C) Acceptance Filter 11 ID Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANMKSEL11[2-0] | | | CANFID113[4-0] | | | | |
| WR | CANMKSEL11[2-0] | | | CANFID113[4-0] | | | | |

CANMKSEL11 Acceptance Filter 11's Mask Selection

Each active Acceptance Filter must cooperate with an Acceptance Filter Mask to decide the acceptance receive ID range. The Acceptance Filter can choose one Acceptance Filter Masks by CANMKSEL11[2-0].

CANMKSEL11[2-0] = 000: Acceptance Filter Mask 0 is selected
CANMKSEL11[2-0] = 001: Acceptance Filter Mask 1 is selected
CANMKSEL11[2-0] = 010: Acceptance Filter Mask 2 is selected
CANMKSEL11[2-0] = 011: Acceptance Filter Mask 3 is selected
CANMKSEL11[2-0] = 100: Acceptance Filter Mask 4 is selected
CANMKSEL11[2-0] = 101: Acceptance Filter Mask 5 is selected
CANMKSEL11[2-0] = 110: Acceptance Filter Mask 6 is selected
CANMKSEL11[2-0] = 111: Acceptance Filter Mask 7 is selected

**CANAFID112 (0x4D) Acceptance Filter 11 ID Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID112[7-0] | | | | | | | |
| WR | CANFID112[7-0] | | | | | | | |

**CANAFID111 (0x4E) Acceptance Filter 11 ID Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID111[7-0] | | | | | | | |
| WR | CANFID111[7-0] | | | | | | | |

**CANAFID110 (0x4F) Acceptance Filter 11 ID Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFID110[7-0] | | | | | | | |
| WR | CANFID110[7-0] | | | | | | | |

**CANAFMK03 (0x50) Acceptance Filter Mask 0 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK03[7-0] | | | | | | | |
| WR | CANFMK03[7-0] | | | | | | | |

**CANAFID02 (0x51) Acceptance Filter Mask 0 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK02[7-0] | | | | | | | |
| WR | CANFMK02[7-0] | | | | | | | |

**CANAFMK01 (0x52) Acceptance Filter Mask 0 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK01[7-0] | | | | | | | |
| WR | CANFMK01[7-0] | | | | | | | |

**CANAFMK00 (0x53) Acceptance Filter Mask 0 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK00[7-0] | | | | | | | |
| WR | CANFMK00[7-0] | | | | | | | |

**CANAFMK13 (0x54) Acceptance Filter Mask 1 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK13[7-0] | | | | | | | |
| WR | CANFMK13[7-0] | | | | | | | |

**CANAFID12 (0x55) Acceptance Filter Mask 1 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK12[7-0] | | | | | | | |
| WR | CANFMK12[7-0] | | | | | | | |

**CANAFMK11 (0x56) Acceptance Filter Mask 1 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK11[7-0] | | | | | | | |
| WR | CANFMK11[7-0] | | | | | | | |

**CANAFMK10 (0x57) Acceptance Filter Mask 1 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK10[7-0] | | | | | | | |
| WR | CANFMK10[7-0] | | | | | | | |

**CANAFMK23 (0x58) Acceptance Filter Mask 2 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK23[7-0] | | | | | | | |
| WR | CANFMK23[7-0] | | | | | | | |

**CANAFID22 (0x59) Acceptance Filter Mask 2 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK22[7-0] | | | | | | | |
| WR | CANFMK22[7-0] | | | | | | | |

**CANAFMK21 (0x5A) Acceptance Filter Mask 2 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK21[7-0] | | | | | | | |
| WR | CANFMK21[7-0] | | | | | | | |

**CANAFMK20 (0x5B) Acceptance Filter Mask 2 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK20[7-0] | | | | | | | |
| WR | CANFMK20[7-0] | | | | | | | |

**CANAFMK33 (0x5C) Acceptance Filter Mask 3 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK33[7-0] | | | | | | | |
| WR | CANFMK33[7-0] | | | | | | | |

**CANAFID32 (0x5D) Acceptance Filter Mask 3 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK32[7-0] | | | | | | | |
| WR | CANFMK32[7-0] | | | | | | | |

**CANAFMK31 (0x5E) Acceptance Filter Mask 3 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK31[7-0] | | | | | | | |
| WR | CANFMK31[7-0] | | | | | | | |

**CANAFMK30 (0x5F) Acceptance Filter Mask 3 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK30[7-0] | | | | | | | |
| WR | CANFMK30[7-0] | | | | | | | |

**CANAFMK43 (0x60) Acceptance Filter Mask 4 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK43[7-0] | | | | | | | |
| WR | CANFMK43[7-0] | | | | | | | |

**CANAFID42 (0x61) Acceptance Filter Mask 4 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK42[7-0] | | | | | | | |
| WR | CANFMK42[7-0] | | | | | | | |

**CANAFMK41 (0x62) Acceptance Filter Mask 4 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK41[7-0] | | | | | | | |
| WR | CANFMK41[7-0] | | | | | | | |

**CANAFMK40 (0x63) Acceptance Filter Mask 4 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK40[7-0] | | | | | | | |
| WR | CANFMK40[7-0] | | | | | | | |

**CANAFMK53 (0x64) Acceptance Filter Mask 5 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK53[7-0] | | | | | | | |
| WR | CANFMK53[7-0] | | | | | | | |

**CANAFID52 (0x65) Acceptance Filter Mask 5 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK52[7-0] | | | | | | | |
| WR | CANFMK52[7-0] | | | | | | | |

**CANAFMK51 (0x66) Acceptance Filter Mask 5 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK51[7-0] | | | | | | | |
| WR | CANFMK51[7-0] | | | | | | | |

**CANAFMK50 (0x67) Acceptance Filter Mask 5 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK50[7-0] | | | | | | | |
| WR | CANFMK50[7-0] | | | | | | | |

**CANAFMK63 (0x68) Acceptance Filter Mask 6 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK63[7-0] | | | | | | | |
| WR | CANFMK63[7-0] | | | | | | | |

**CANAFID62 (0x69) Acceptance Filter Mask 6 Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK62[7-0] | | | | | | | |
| WR | CANFMK62[7-0] | | | | | | | |

**CANAFMK61 (0x6A) Acceptance Filter Mask 6 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK61[7-0] | | | | | | | |
| WR | CANFMK61[7-0] | | | | | | | |

**CANAFMK60 (0x6B) Acceptance Filter Mask 6 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK60[7-0] | | | | | | | |
| WR | CANFMK60[7-0] | | | | | | | |

**CANAFMK73 (0x6C) Acceptance Filter Mask 7 Byte 3 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK73[7-0] | | | | | | | |
| WR | CANFMK73[7-0] | | | | | | | |

**CANAFID72 (0x6D) Acceptance Filter 7 Mask Byte 2 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK72[7-0] | | | | | | | |
| WR | CANFMK72[7-0] | | | | | | | |

**CANAFMK71 (0x6E) Acceptance Filter Mask 7 Byte 1 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK71[7-0] | | | | | | | |
| WR | CANFMK71[7-0] | | | | | | | |

**CANAFMK70 (0x6F) Acceptance Filter Mask 7 Byte 0 RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CANFMK70[7-0] | | | | | | | |
| WR | CANFMK70[7-0] | | | | | | | |

## 1.8 Basic Operations

The basic function of CAN controller will be discussed in this section. The following chapter will describe the advanced functions of CAN controller. To communicate with other nodes on the CAN-bus, CAN controller must be configured properly as described.

### 1.8.1 Setting of Operation Mode

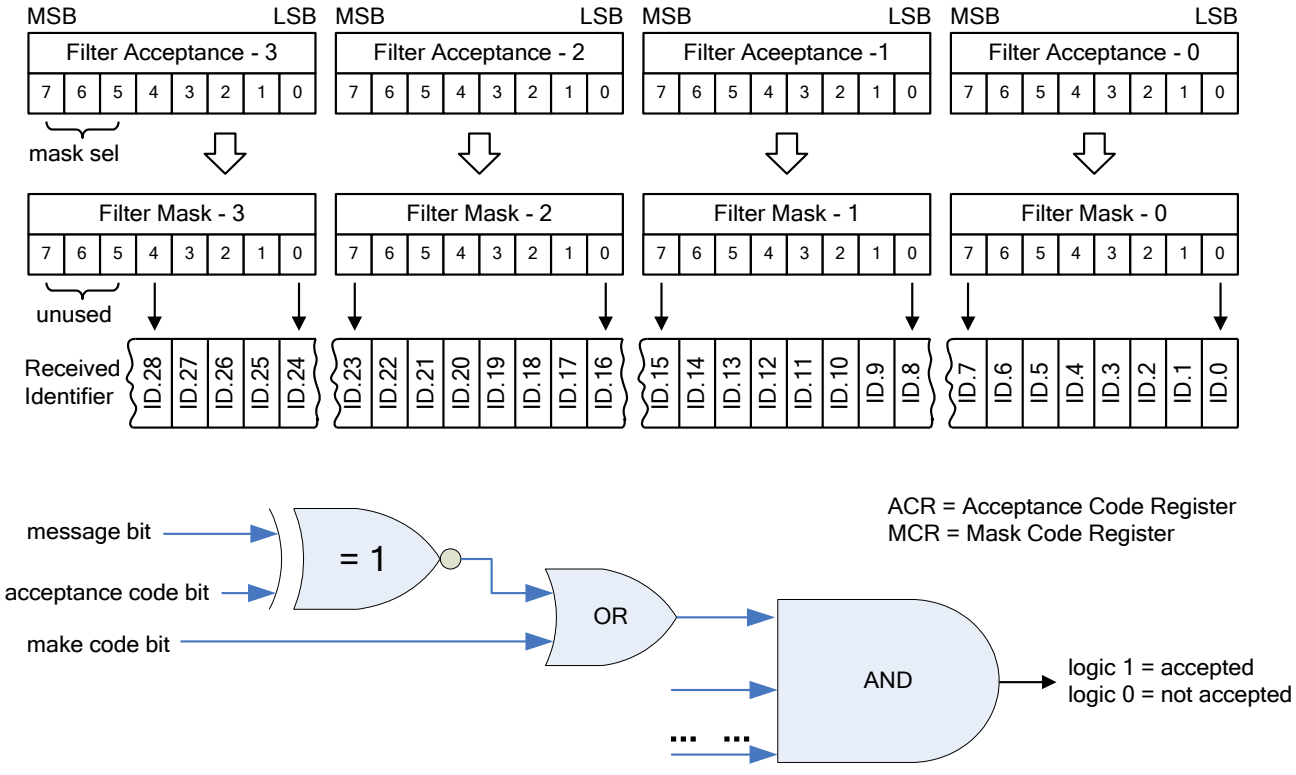
Some registers are writable only in Reset mode such as Bus Timing registers, so we must write these registers in the Reset mode and the CAN controller must quit Reset mode when these registers are configured properly to make the controller work.

### 1.8.2 Baud Rate Setting

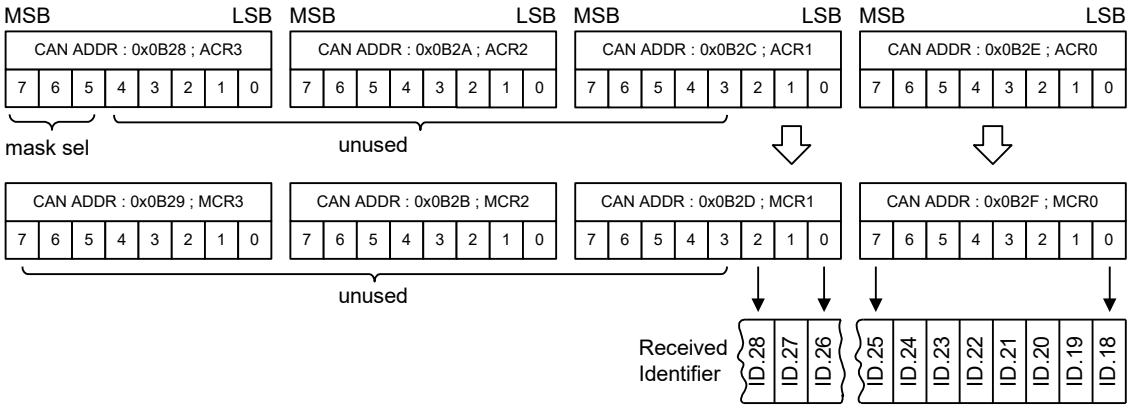
The baud rate in the CAN-bus is determined by the two registers CANBTR0 and CANBTR1. The following example explains how to calculate the baud rate of 500K assuming system clock is 16MHz, and CANBTR0=01h CANBTR1=14h. Then the CANCLK is SYSCLK/2/(CANCS[5-0]+1) = SYSCLK/4 = 4MHz. Thus each time quanta is 250nsec. Then the one CAN bit time BT is (1 + (TSEG1+1) + (TSEG2+1)) x 250 nsec = (1 + 2 + 5) x 250nsec = 2usec. As the result the CAN bus baud rate is 500Kbps.

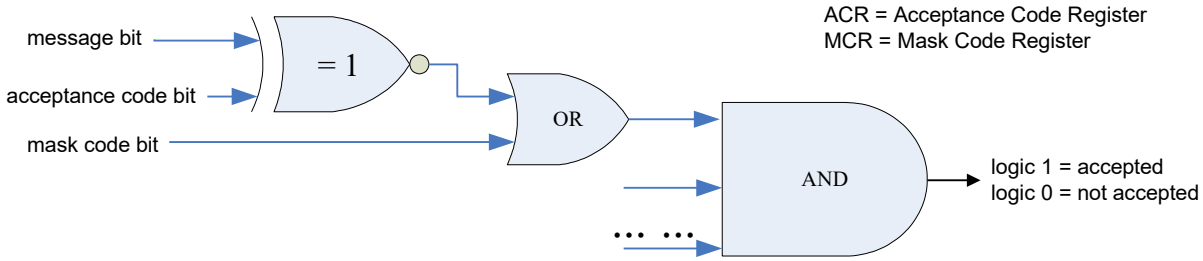
### 1.8.3 Setting of Acceptance Filter

With the help of the acceptance filter, the CAN controller can receive proper frames from CAN-bus only when the identifier bits of the received frames are equal to the predefined ones within the acceptance filter registers. The acceptance filter is defined by the Acceptance Filter Code Registers (ACR) and the Acceptance Filter Mask Code Registers (MCR). The bit patterns of frames to be received are defined within the Acceptance Filter Code registers. The corresponding mask code registers allow define certain bit positions to be 'don't care'. After hardware reset, all the Acceptance Filter Code registers and Acceptance Filter Mask Code registers are cleared.
In IS31IO8972, there are 12 groups of acceptance filter and 8 groups of acceptance filter mask. We must enable at least one group of filter with a corresponding acceptance filter mask previously. Each group of filter has 4-byte Acceptance Filter Code and 4-byte Acceptance Filter Mask Code. We take Acceptance Filter 0 in CAN as an example: the address of the Acceptance Filter Code 3 is 0x20, the address of the Acceptance Filter Code 2 is 0x21 and so on. If the Acceptance Filter 0 selects Acceptance Filter Mask 1 as its corresponding mask code by writing a '001' into CANMKSEL0[2-0], the corresponding Acceptance Filter Mask is located at address 0x54 to 0x57. When received message is standard format frame, the lower 11 bits of the Acceptance Filter Code and the Acceptance Filter Mask Code are valid. When received message is extended format frame, the lower 29 bits of the Acceptance Filter Code and the Acceptance Filter Mask Code are
valid.




Extended frame: if an extended format frame is received, the complete identifier including the RTR bit is used for acceptance filtering. For a successful reception of a message, all single bit comparisons have to signal acceptance. It should be noted that the 3 most significant bits of ACR are Acceptance Filter Mask selection and the 3 most significant bits of MCR3 are not used.



Standard frame: if a standard format frame message is received, the completely identifier are used for acceptance filtering. For a successful reception of a frame, all single bit comparisons have to signal acceptance. The 3 most significant bits of ACR are Acceptance Filter Mask selection. The ACR3[4-0], MCR3, ACR2, MCR2 and the 5 most significant bits of ACR1 and MCR1 are not used. In order to be compatible with future products, these unused bits in Mask Code registers should be programmed to be 'don't care' by setting these bits to logic 1.

### 1.8.4 Transmitting a Frame

You must write transmit buffer to send a frame to CAN-bus, then set the transmit request bit to logic 1. The controller will wait for the bus idle and sends the frame to the bus. For example, CAN Controller will send a standard format frame to CAN-bus from the transmit buffer 0. Frame identifier is 0x001, the data have a length of 8 and the data are 11h, 22h, 33h, 44h, 55h, 66h, 77h and 88h.The program written in Keil-C is as follows:

*1. Write_reg(0x80,0x00); // disable Auto-dispatch function*
*2. Write_reg(0x81,0x08); // write the frame information register, FF=0, RTR=0, DLC=8*
*3. Write_reg(0x82,0x00); // write the TX ID3 Register*
*4. Write_reg(0x83,0x00); // write the TX ID2 Register*
*5. Write_reg(0x84,0x00); // write the TX ID1 Register*
*6. Write_reg(0x85,0x01); // write the TX ID0 Register*
*7. Write_reg(0x86,0x11); // write the TX data 1 Register*
*8. Write_reg(0x87,0x22); // write the TX data 2 Register*
*9. Write_reg(0x88,0x33); // write the TX data 3 Register*
*10. Write_reg(0x89,0x44); // write the TX data 4 Register*
*11. Write_reg(0x8A,0x55); // write the TX data 5 Register*
*12. Write_reg(0x8B,0x66); // write the TX data 6 Register*
*13. Write_reg(0x8C,0x77); // write the TX data 7 Register*
*14. Write_reg(0x8D,0x88); // write the TX data 8 Register*
*15. Write_reg(0x05,0x01); // write the transmit request bit*

*Note that the prototype of the function Write_reg is:*
*Void Write_reg(unsigned int address, unsigned char value );*

### 1.8.5 Receiving a Frame

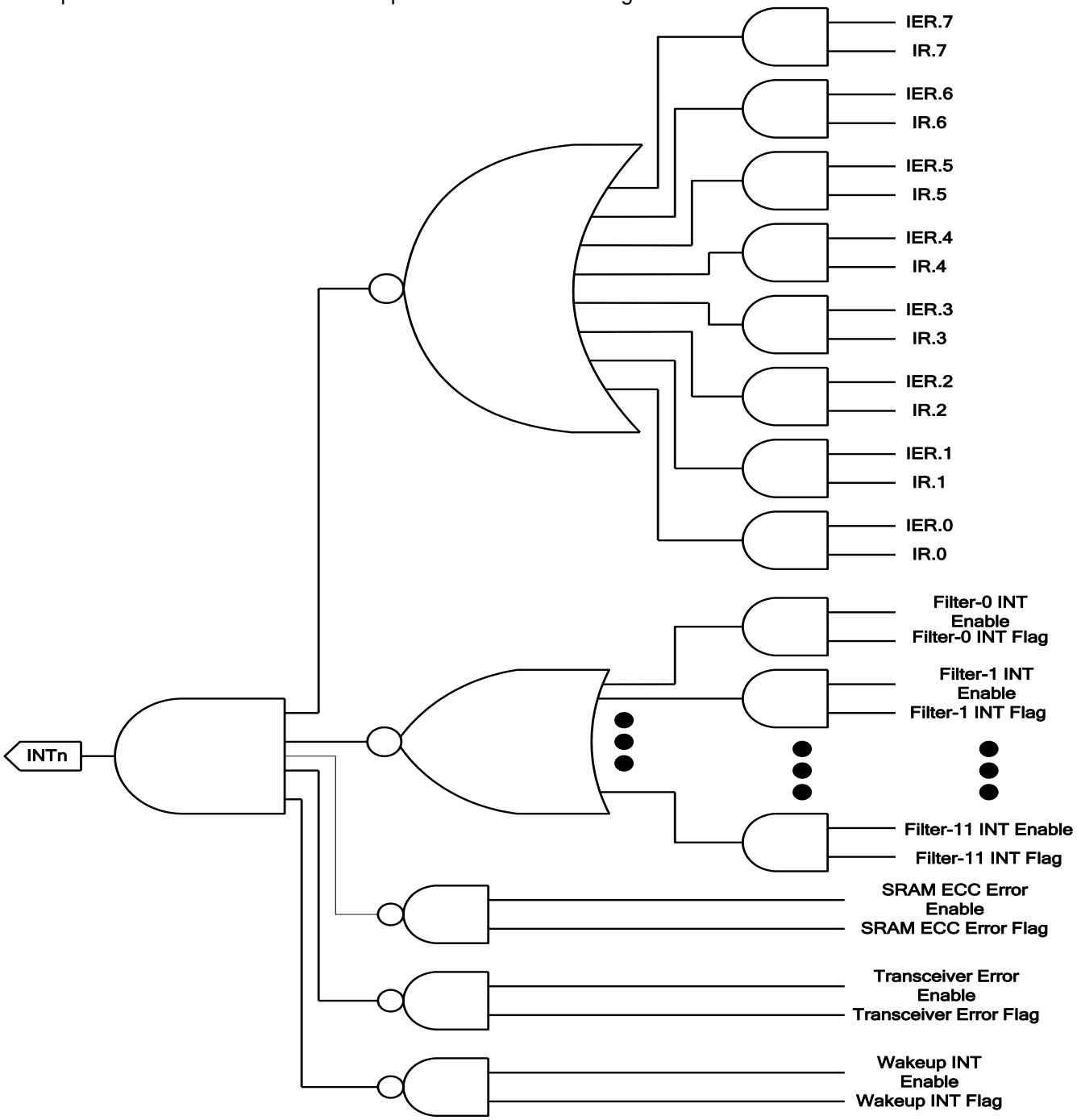
When CAN controller receives a frame successfully and the receive interrupt is enabled, the receive interrupt bit will be set logic 1. MCU is informed by the receive interrupt and reads the data in the receive buffer out, then releases the receive buffer. Please pay special attention here that the data must be read before releasing receive buffer, because the value in the receive buffer has changed after releasing receive buffer.

### 1.8.6 Dispatching

The CAN controller can transmit data periodically to CAN-bus to report its status. To realize this function, we must configure the dispatch time registers properly. IS31IO8972 has two Dispatch Interval Timers, D0TR0 and D0TR1. Both D0TR0 and D0TR1 are 16-bit dispatch timers. A non-zero value must be programed into the choice dispatch interval timer to enable the timer. The 16 dispatch time bits control the period that the controller transmits data to CAN-bus and the dispatch period is calculate as follows: Provided the bit period is $t_b = \frac{1}{BaudRate}$ and the value of the 16 dispatch time bits is "A", then the dispatch time is "100 x (A+16) x $t_b$" second. In other words, the CAN controller transmits a message from the active TX buffers to the CAN-bus every "100 x (A+16) x $t_b$" second. When dispatch time bits are all set to logic 0, the dispatch function is disabled. If Auto-Dispatch Mode Select (ADMOD) is cleared, when bus error in dispatching a frame, it can transmit an error frame and can't transmit the failed frame again.

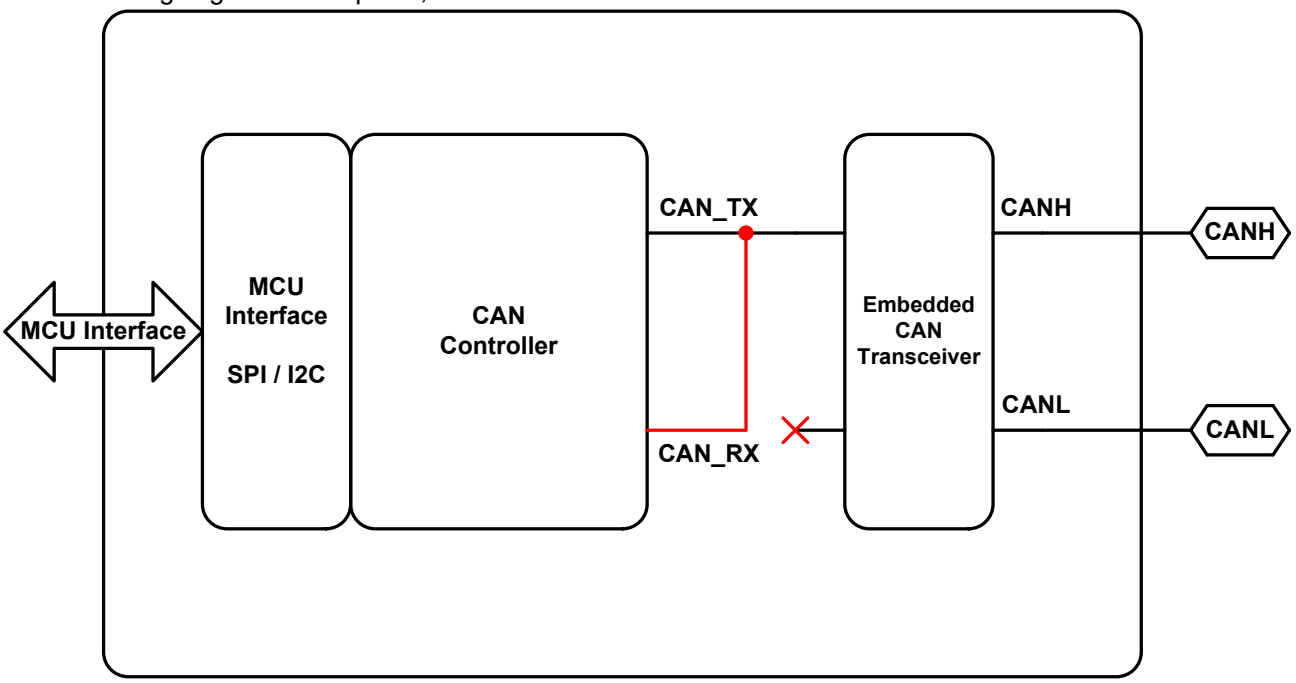
## 1.9 Interrupt

In IS31IO8972, the INTn output pin source from CAN Interrupt Flag (CANINTF), CAN Acceptance Filter Match Interrupt Flag (CANRFIF), and CAN Extra Interrupt Flag (CANEXIF). Each interrupt flag bit has a corresponding interrupt enable bit. The function of INTn pin is described as the figurate below.



## 1.10 Self Test Mode

The self test mode is of the CAN control is the same normal operation mode except when the message is sent, the ACK error is ignored. This allows the software to initiate a “self reception request” command. Upon this command, a message is transmitted to the bus and will be treated as successfully transmitted without check the ACK flag. The transmitted message is also received by the sending CAN controller if the ID field matches with the acceptance filter. The software can then check the correctness of the received message and perform loop-back test. The loop-back can be done by connecting CANH and CANL with a terminal resistor, or can be done through the external transceiver on the actual CAN bus, or setting the Self-Loop Mode. When loop back from the bus, the CAN bus needs to be properly terminated to ensure correct electrical level for normal bus operation.
The following figurate illustrates the internal connection when CAN is in Self-Loop Mode. Under Self-loop Mode, the internal signal ‘CAN_RX’ disconnects from the embedded Transceiver and is short with ‘CAN_TX’, Self-Loop test usually goes on in self test mode (STE) with self receive request (SRR) command to verify the controller whether it works. When going on self-loop test, the controller can’t receive data from CAN-bus.




To enable CAN Self-Loop Test Mode, MCU must unlock CAN Test Locker Register by writing an “A6H” into the least four bits of CAN Test Locker Register at address 0x7E. Then enable the Self-Loop Test Mode by setting the CAN Self-Loop Test Enable bit at CAN Test Enable Register (address 0x7F).

**CANPTLOCK (0x7F) CAN Protection Locker Register WO (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | | | | | | | |
| WR | CANPTLOCK[7-0] | | | | | | | |

CAN Protection Locker Register is a security register to avoid an erroneous register programming. It emulates a ticket that must be purchased before modifying a critical register. MCU must program an “0xA6” into the Protection Locker Register at address 0x7F to unlock the security before to program the protected registers, including CAN Test Enable Register (0x7E), Regulator Configuration Register (0x77) and I/O Control Register (0x78). Protection Locker Register does not limit the read access of those protected registers. The security bit will be locked again once MCU programs any data into any address. This register is a write only register.

CANPTLOCK CAN Protection Locker Register
Write '0xA6' into CANPTLOCK to unlock the security
Write a not '0xA6' value into CANPTLOCK or write other addresses to lock the security again.

**CANTESTEN (0x7E) CAN Test Enable Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | - | CANFTXD | | - | CANMBIST | - | CANSLPE |
| WR | CANRST | - | CANFTXD | | - | CANMBIST | - | CANSLPE |

CAN Test Enable Register includes four test signals and is protected by CANPTLOCK. Before program CANTESTEN, the CANPTCLOCK must be unlocked by writing '0xA6'. In order to avoid the fault, only one of 'CANRST', 'CANDTE', 'CANMBIST' and 'CANSLPE' bit can be set at the same time.

CANRST CAN Reset
Write 1 to generate a reset pulse to reinitiate whole chip. This bit is cleared automatically by hardware. CANRST is a write-only bit.

CANFTXD Force CAN dominant status
CAMFTXD=1: IS31IO8972 drives the 'dominant' bit on CANH and CANL until this bit is cleared by software. This test function is only valid when both TSE and RSTM were set.

CANMBIST CAN SRAM Built-in Self Test Enable
This bit is reserved for the engineer test. User must always keep CANMBIST=0 when program CANTESTEN register.

CANSLPE CAN Self-Loop Test Mode
CANSLPE=1: Enable CAN Self-Loop Test Mode
CANSLPE=0: Disable CAN Self-Loop Test Mode

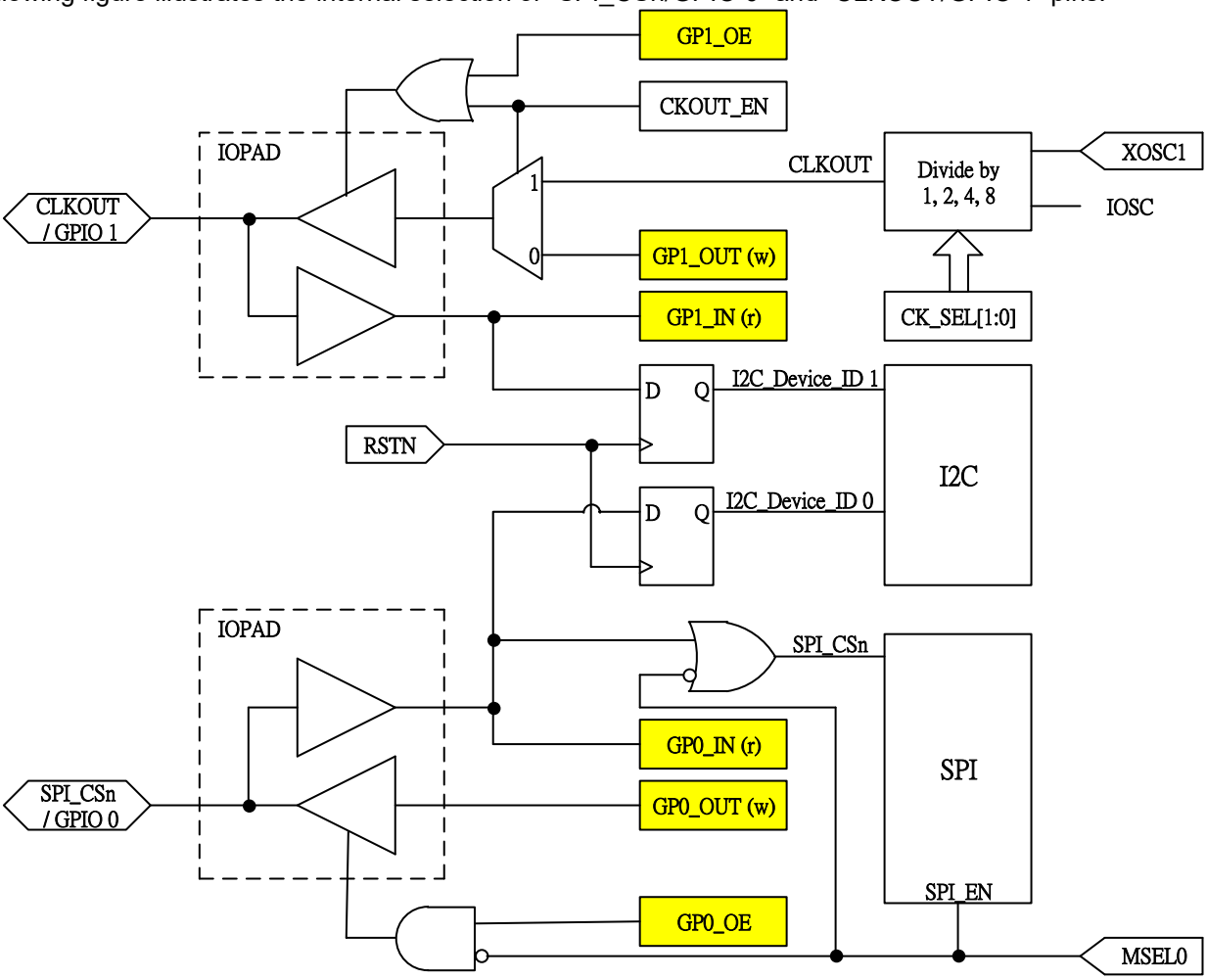
## 2. I/O CONFIGURATION

In IS31IO8972, there are two general-purpose input/output pads with multi-function. When SPI interface was selected (MSEL0 = 1), “SPI_CSn/GPIO 0” is SPI chip select input with low active. When deselect SPI interface (MSEL = 0), “SPI_CSn/GPIO 0” is a general-purpose input/output pin and controlled by GPIO Control Register at address 0x71.

In the initial setting, “CLKOUT/GPIO 1” is a clock output pin. The clock output pin is provided to the system designers for use as the main system clock or as a clock input for other devices. The CLKOUT has an internal prescaler that can divide the frequency of XOSC1 by 1, 2, 4, and 8. The CLKOUT function is controlled and the prescaler is selected via the CLKOUT Control Register at address 0x78. IS31IO8972 enters stop mode by setting Stop Mode bit. Before turn off all clock sources, IS31IO8972 will drive sixteen additional clock cycles on “CLKOUT/GPIO 1” pin when the Stop Mode bit is set. In stop mode, the idle state of “CLKOUT/GPIO 1” is low. Users can select “CLKOUT/GPIO 1” as a general-purpose input/output pin by setting the CLKOUT Enable to zero. When CLKOUT Enable is clear, GPIO Control Register will take over “CLKOUT/GPIO 1” pin.

“SPI_CSn/GPIO 0” and “CLKOUT/GPIO 1” are also used to decide the least two ID address of slave I²C. During the power-on initial state, IS31IO8972 captures the wire level of “SPI_CSn/GPIO 0” and “CLKOUT/GPIO 1” at the rising edge of RSTN input as I²C’s ID-0 and ID-1. System designers must pull-up or pull-down these two pins as the I²C ID[1:0] in system boards.

The following figure illustrates the internal selection of “SPI_CSn/GPIO 0” and “CLKOUT/GPIO 1” pins.




**I/O Selection of SPI_CSn/GPIO0 and CLKOUT/GPIO 1**

## 2.1 Registers Description

**IOCTL (0x78) I/O Control Register RW (0x86)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | CKOEN | CKOSEL[1] | CKOSEL[0] | - | OSCSEL | XOSCEN | IOSCEN | STOP |
| WR | CKOEN | CKOSEL[1] | CKOSEL[0] | - | OSCSEL | XOSCEN | IOSCEN | STOP |

I/O Control Register is a Protection Locker Register protected register. MUC must unlock the security bit by programming an “A6H” into Protection Locker Register before program the register. In order to avoid CAN bus fault, OSCESL, XOSCEN, IOSCEN and STOP are programmable under CAN Reset Mode.

CKOEN — CKOEN=1: Enable Clock Output Function on “CLKOUT/GPIO 1” pin. The output frequency depends on CKOSEL[1-0].
CKOEN=0: Disable Clock Output Function on “CLKOUT/GPIO 1” pin. The “CLKOUT GPIO 1” is a general-purpose input/output pin and CKOSEL[1-0] is ignored.

CKOSEL[1-0] — Clock Output Frequency Selection
CKOSEL[1-0]=00: Clock output frequency = system clock frequency
CKOSEL[1-0]=01: Clock output frequency = system clock frequency / 2
CKOSEL[1-0]=10: Clock output frequency = system clock frequency / 4
CKOSEL[1-0]=11: Clock output frequency = system clock frequency / 8

OSCSEL — System Clock Source Selection
OSCSEL can be read and written in CAN Reset mode and is read-only in CAN Operating mode. The system clock can be switched only when both IOSC and XOSC were active.
OSCSEL=1: XOSC is selected as system clock source
OSCSEL=0: IOSC is selected as system clock source

XOSCEN — XOSC Enable
XOSCEN can be read and written in CAN Reset mode and is read-only in CAN Operating mode. Clear this bit will turn off XOSC immediately if IOSC was selected as system clock. XOSC cannot be disabled if XOSC is selected as the system clock source. Setting Stop mode, XOSC will be disable automatically after drive 16 CLKOUT pulse. XOSC will be re-active by hardware when leave Stop mode by a resume trigger no matter the XOSC enable status before enter Stop mode.
XOSCEN=1: Enable XOSC.
XOSCEN=0: Disable XOSC.

IOSCEN — IOSC Enable
IOSCEN can be read and written in CAN Reset mode and is read-only in CAN Operating mode. Clear this bit will turn off IOSC immediately if XOSC was selected as system clock. IOSC cannot be disabled if IOSC is selected as the system clock source. Setting Stop mode, IOSC will be disable automatically after drive 16 CLKOUT pulse. IOSC will be re-active by hardware when leave Stop mode by a resume trigger no matter the IOSC enable status before enter Stop mode.
IOSCEN=1: Enable IOSC.
IOSCEN=0: Disable IOSC.

STOP — System Stop Mode
STOP Mode bit can be read and written in CAN Reset mode and is read-only in CAN Operating mode.
STOP=1: Enter Stop Mode. IS31IO8972 will turn off XOSC and IOSC then enter power saving mode. The CLKOUT will drive 16 extra clock pulses before turn off the system clock. An external wakeup trigger from SPI/ I²C or CAN bus will wakeup the system and turn on the XOSC and IOSC. Writing a ‘zero’ into this bit is invalid.

**GPIOCTL (0x79) GPIO Control Register RW (0x00)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | - | - | GP1OE | GP0OE | - | - | GP1IN | GP0IN |
| WR | - | - | GP1OE | GP0OE | - | - | GP1OUT | GP0OUT |

IS31IO8972 has two general purpose I/O pins, GPIO0 and GPIO1. GPIO0 is combined with SPI_CSn. If SPI was active by setting MSEL0=1, the GPIO0 will be occupied by SPI and lost the GPIO function. GPIO1 is combined with clock output function. Before active the GPIO function of GPIO1, the clock output function must be disabled by clear CKONE.

GP1OE GPIO1 Output Enable
GP1OE=1: GPIO1 is an output pad. The output value is GP1OUT.
GP1OE=0: GPIO1 is an input pad. The wire level will be latched into GP1IN.

GP1OE GPIO1 Output Enable
GP1OE=1: GPIO1 is an output pad. The output value is GP1OUT.
GP1OE=0: GPIO1 is an input pad. The wire level will be latched into GP1IN.

GP1IN GPIO1 Input Data
This is a read-only bit. When CKOEN=0 and GP1OE=0, the wire level of GPIO1 can be checked out by reading GP1IN register.

GP1OUT GPIO1 Output Data
This is a write-only bit. When CKOEN=0 and GP1OE=1, GP1OUT decides the wire level of GPIO1.

GP00N GPIO0 Input Data
This is a read-only bit. When MSEL0=0 and GP0OE=0, the wire level of GPIO0 can be checked out by reading GP0IN register.

GP0OUT GPIO0 Output Data
This is a write-only bit. When MSEL0=0 and GP0OE=1, GP0OUT decides the wire level of GPIO0.

## 3. MCU INTERFACE

IS31IO8972 provides two popular serial interfaces to link with the external MCU, Serial Peripheral Interface (SPI) and I²C bus. Only one serial interface can be active in an application system. Pull up the “MSEL0” input pad to enable SPI interface and disable I²C interface. Pull down both “MSEL0” and “SPI_SI / MSEL1” input pads to enable I²C interface. In IS31IO8972, both SPI and I²C are slave controllers. They receive and respond the commands from MCU by selected serial interface. There are more detail description about the SPI can I²C command in the following sections.

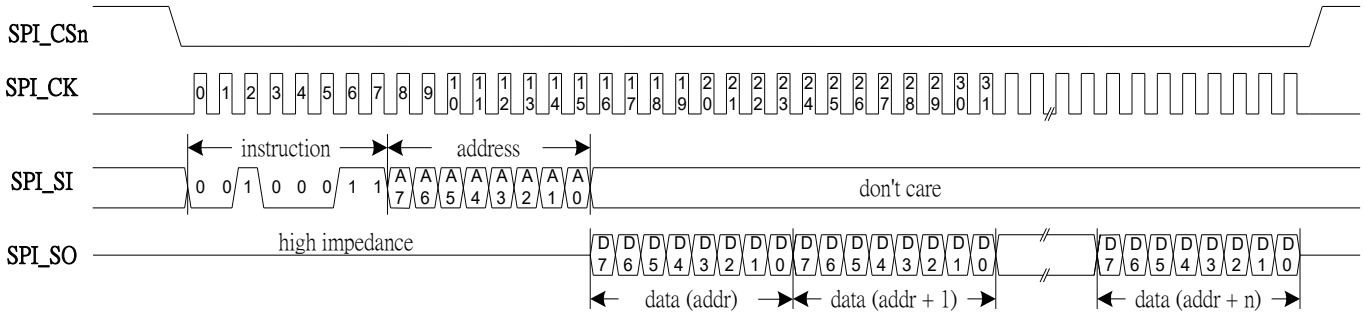
### 3.1 SPI

The Serial Peripheral Interface (SPI) is an enhanced synchronous serial hardware, which is compatible with Motorola’s SPI specifications. In IS31IO8972, only a slave SPI controller is implemented to receive and respond the commands from MCU. The “SPI_CSn” is input only pad to identify the start and end of a command/data frame. The “SPI_CK” is clock input pad and its maximal bit rate is 3Mbps under 24MHz system clock. The “SPI_SI” is command/data input pad and the “SPI_SO” is data output pad.

IS31IO8972 only supports SPI mode 0, 0 (CPOL = 0, CPHA = 0) and 1, 1 (CPOL = 1, CPHA = 1). Data always is latched at SPI_CK’s rising edge and shifted out at SPI_CK’s falling edge in these two modes. The command and data are MSB-first. After the falling edge of SPI_CSn, the first byte received from SPI_SI is the “Instruction Byte” and the following byte is the “Address Byte”. IS31IO8972 supports three instructions to access the CAN controller, Read, Write and Bit Modify.

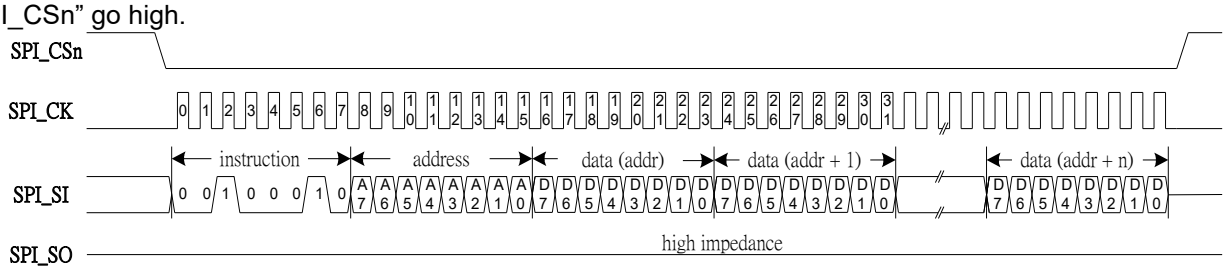
| Instruction | Format | Description |
|---|---|---|
| Read | 0010 0011 | Read data from register beginning at selected address |
| Write | 0010 0010 | Write data to register beginning at selected address |
| Bit Modify | 0010 0101 | Bit modify selected register |

The following figure illustrates the data read from registers beginning at selected address. The first byte is the instruction “0010 0011” and the second byte is the start address of accessed register. The following byte shift-out from “SPI_SO” is the data of selected register. After the first shift-out data, the next shifting data is the data of “address+1”. In other words, hardware always increases the address by 1 automatically when shifts out one byte data until the “SPI_CSn” go high.




**Byte Read by SPI**

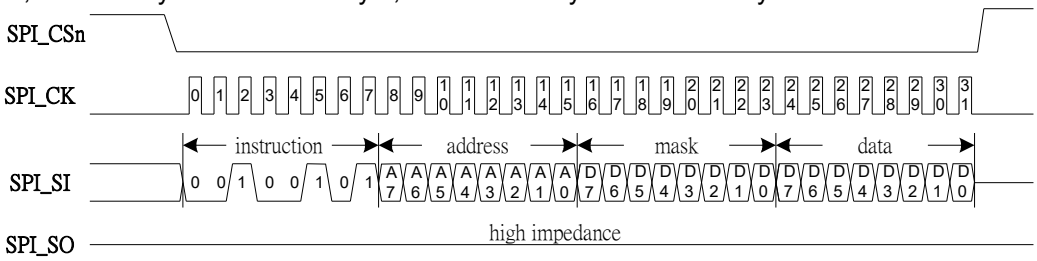
The following figure illustrates the data write to registers beginning at selected address. The first byte is the instruction “0010 0010” and the second byte is the start address of accessed register. The following byte shift-in from “SPI_SI” is the data to be written into selected register. After the first shift-in data, the next shifting data is the data for “address+1” register. In other words, hardware always increases the address by 1 automatically when shifts in one byte data until the “SPI_CSn” go high.




**Byte Write by SPI**

The following figure illustrates the bit modify by SPI host. The “Bit Modify” instruction only can access single byte. This

instruction includes four bytes. The first byte is the instruction “0010 0101” and the second byte is the address of accessed register, the third byte is the mask byte, and the last byte is the data byte.




**Bit Modify by SPI**

The mask byte determines which bit in the selected register will be allowed to change. A ‘1’ in the mask byte will allow a bit in the selected register to be modify and a ‘0’ will not. The data byte determines what value the modified bits in the selected register will be changed to. A ‘1’ in the data byte will set the bit and a ‘0’ will clear the bit, provided that the mask for that bit is set to a ‘1’.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mask Byte | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Data Byte | 1 | 1 | 0 | 0 | X | X | X | X |
| Original Register Contents | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| Upated Register Contents | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

Note : X is don't care, can be 0 or 1

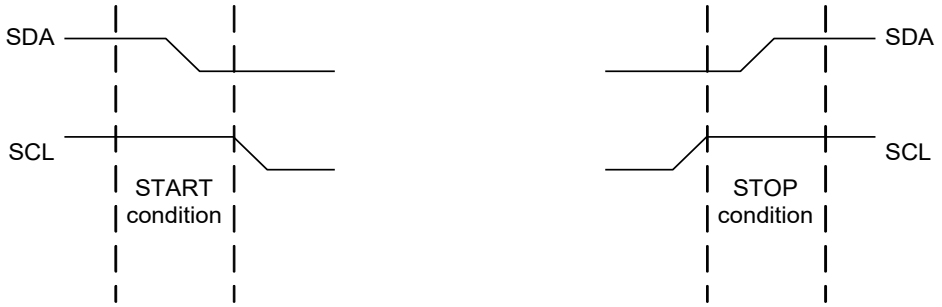
**True Table of Bit Modify**

## 3.2 Slave I²C

IS31IO8972 provides a slave I²C interface to connect with MCU if both “MSEL0” and “SPI_SI / MSEL1” are tired to low. Two wires, serial clock input (I²C_SCL) and serial data (I²C_SDA), carry information and data between MCU and IS31IO8972. The maximal bit rate is 2Mbps under 24MHz system clock without clock stretching. IS31IO8972 also provides four slave ID address option which selected by “CLKOUT/GPIO 1” and “SPI_CSn/GPIO 0” input pins during the initial state. Following table describe the relation between I/O pad and I²C address.

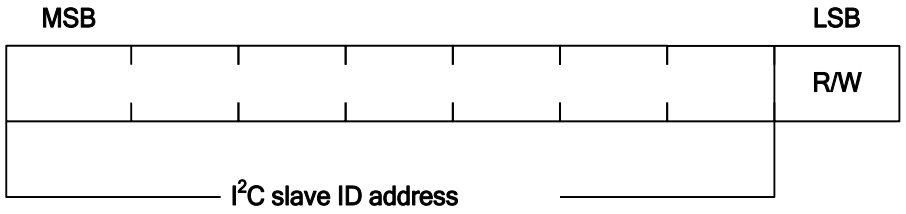
| Slave I²C Address | TMODE | MSEL0 | MSEL1 | GPIO 0 | GPIO 1 |
|---|---|---|---|---|---|
| 0x90 / 0x91 | Tie low | Tie low | Tie low | Pull-low | Pull-low |
| 0x92 / 0x93 | Tie low | Tie low | Tie low | Pull-high | Pull-low |
| 0x94 / 0x95 | Tie low | Tie low | Tie low | Pull-low | Pull-high |
| 0x96 / 0x97 | Tie low | Tie low | Tie low | Pull-high | Pull-high |

*Note: Pull-low or Pull-high must be implemented by the external resistors.*



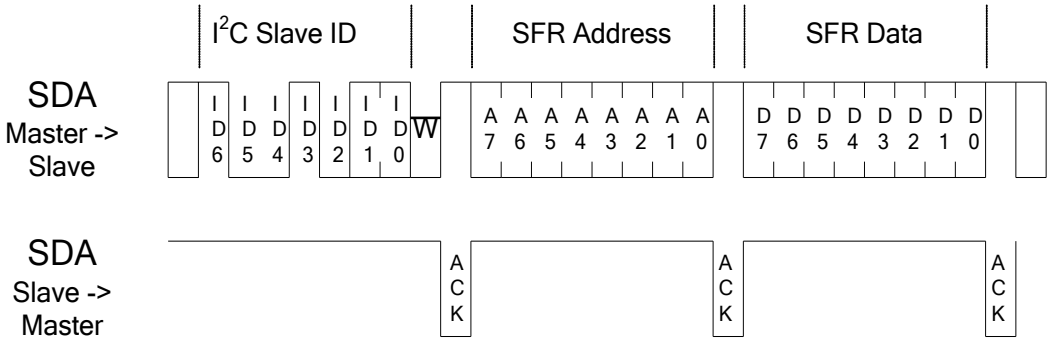

**I²C START and STOP Conditions**

After the I²C START condition transmitted from MCU, the first byte received on I²C bus is “Slave ID Address + Read/Write“. “Slave ID Address” has seven bits length, and the least two bits are decided by “CLKOUT/GPIO 1” and “SPI_CSn/GPIO 0” input pins in the initial state. The last bit is Read/Write command bit. If MCU is going to program data into IS31IO8972 in the following procedure, this “Read/Write” bit shall be tied low. Otherwise MCU can read the designate data from IS31IO8972 on the I²C bus if ties “Read/Write” bit to high.




**I²C First Byte after the START Condition**

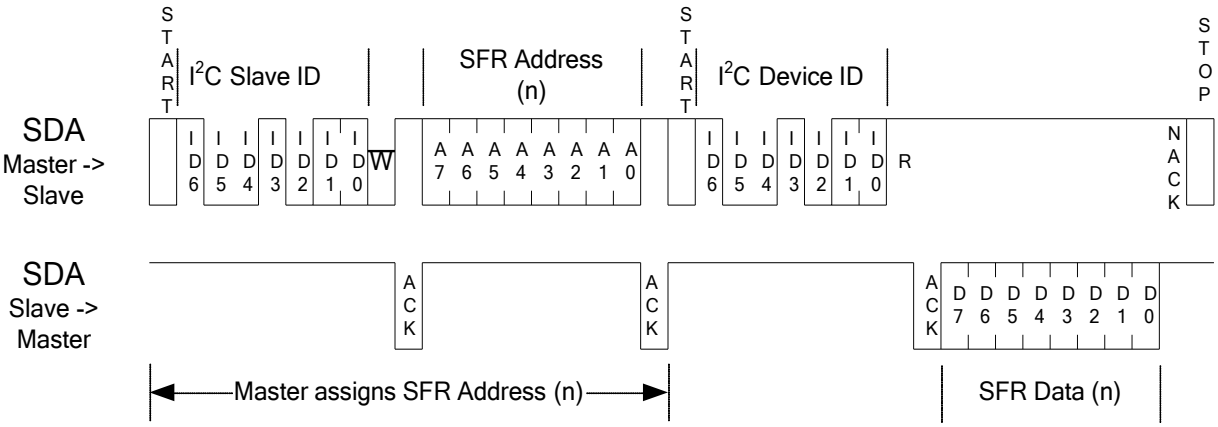
The following figure illustrates the single byte write by I²C master. MCU sends a valid “I²C Slave ID address” and ties “Read/Write” to low in the first byte. After receives an ACK (SDA = 0) at the ninth bit of each byte, MCU sends the designate “SFR Address” and the “SFR Data”.




**Single Byte Write by I²C**

The following figure illustrates the single byte read by I²C master. MCU must issue a write procedure to indicate the

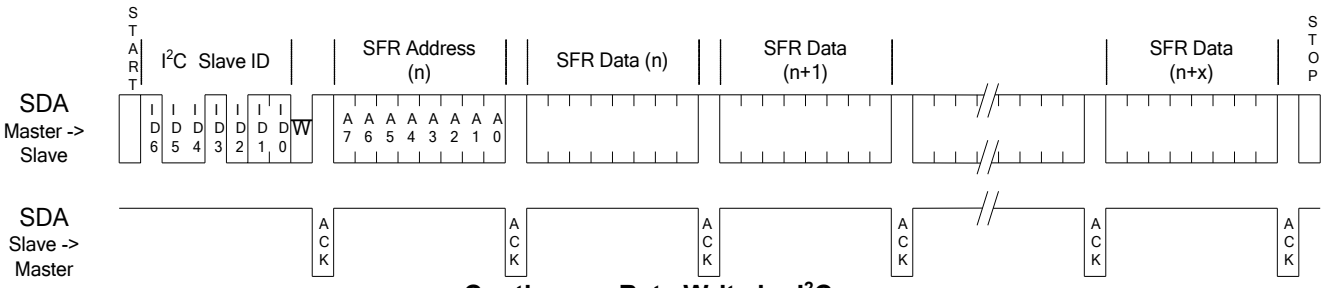
SFR Address that begins to read out the data. Then issue another read procedure by tying high the eighth bit (Read/Write) of the first byte. IS31IO8972 will shift out the selected data in the following byte. MCU must acknowledge every incoming data byte by pulling down the ACK bit except the last byte. MCU must terminate the reading procedure and get the ownership of data bus ($I^2C$_SDA) by pulling up the ACK bit. When doesn't receive an acknowledgement in the ACK bit, IS31IO8972 will release the data line to allow the master to generate a STOP or repeated START condition.



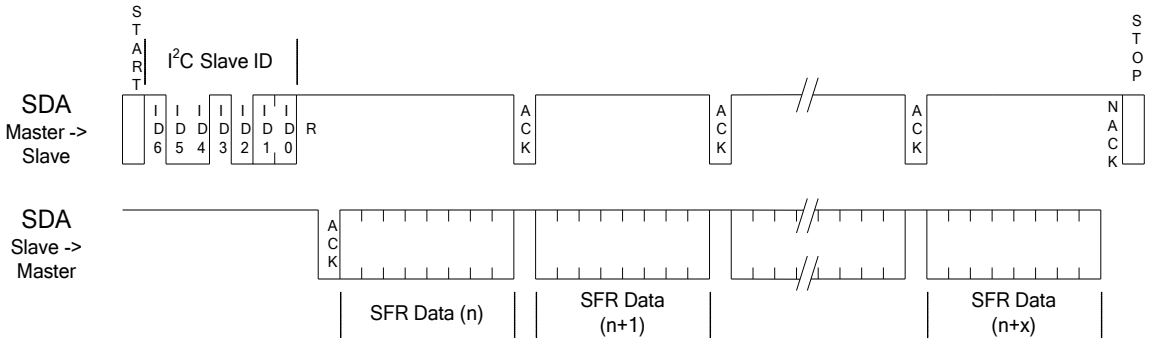

**Single Byte Read by $I^2C$**

The slave $I^2C$ also provides continuous access mode for MCU to read and write multi-byte data. The hardware will increase the address by 1 automatically based on the "Address[7-0]" received in the first byte. MCU can terminate the writing procedure anytime except the ACK bit. During the continuous reading procedure, MCU must terminate the procedure by pulling up the ACK bit and IS31IO8972 releases the data line to allow MCU to generate a STOP or repeated START condition.

After receive a START condition, the last bit of the first byte is "Read/Write". If "Read/Write" is low, the second byte received is "SFR Address". IS31IO8972 will update the internal Address Counter while acknowledge the second incoming byte. IS31IO8972 always increases the internal Address Counter by 1 when receive a byte from Master or transmit a byte to Master.




**Continuous Byte Write by $I^2C$**




**Continuous Byte Read by $I^2C$**

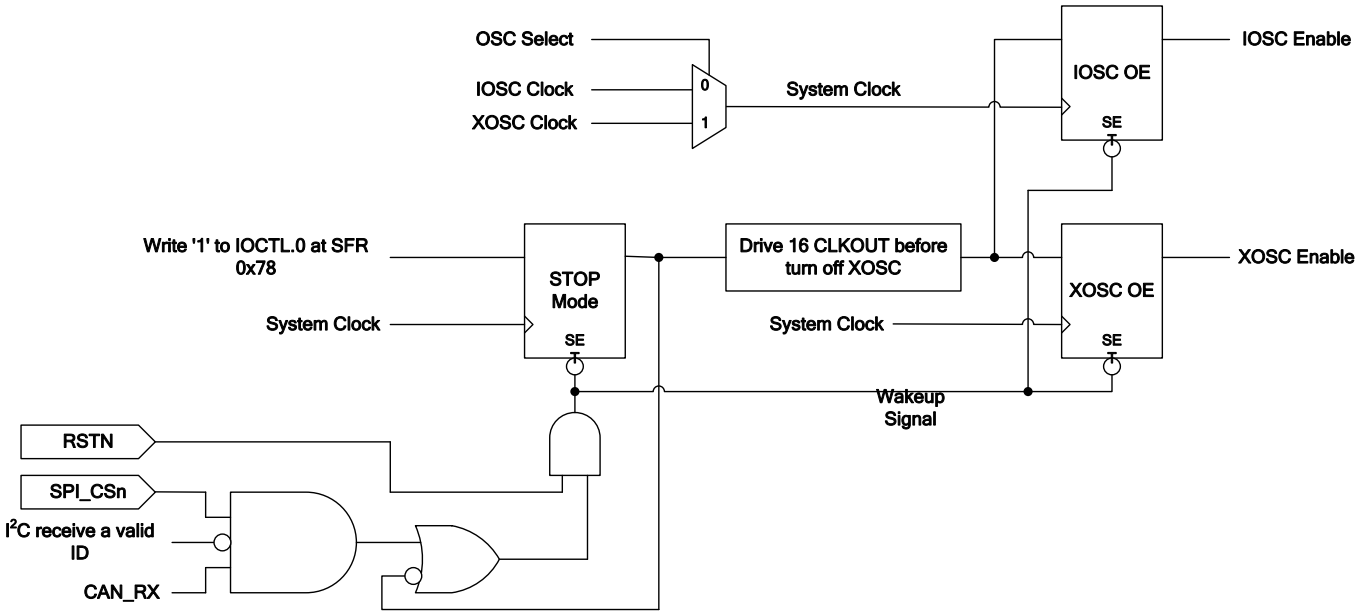
## 4. POWER SAVING MODE

IS31IO8972 provides a power saving mode by turning off the external oscillator (XOSC) and internal oscillator (IOSC). Set the bit-0 of the CLKOUT control register at address 0x78 will enable the STOP mode. Hardware will drive sixteen additional system clock cycles on the CLKOUT output pin then turn off the external oscillator and the internal oscillator. While the oscillators are turned off, whole chip will enter power saving mode. There are three events can wake up the system except re-active the RSTN input pin.

The first event is detecting a 'dominant' bit on CAN bus. When CAN transceiver detects a 'dominant' bit on CAN bus, the transceiver will pull-down the internal wire 'CAN_RX' to inform the CAN controller to leave Stop mode and re-active both the external oscillator and the internal oscillator.

The second wake-up event is detecting a 'falling edge' on "SPI_CSn". SPI host can wake up IS31IO8972 by pulling down the SPI chip select input "SPI_CSn". IS31IO8972 will resume the external oscillator and the internal oscillator when the falling edge is detected. While pull down the "SPI_CSn", Host cannot transmit the first command/data frame by SPI until the oscillator is stable. The stable time is about 10ms.

The third wake-up event is received a valid I²C ID address under STOP mode. IS31IO8972 will wake up from STOP mode when a valid I²C ID address is shifted in after a 'Start' condition. Before shift out next frame, Host must wait at least 10ms for the oscillator stabilization after wake up IS31IO8972 by shifting a valid address in SDA.




**Block Diagram of Power Saving Control**

# 5. CAN Transceiver

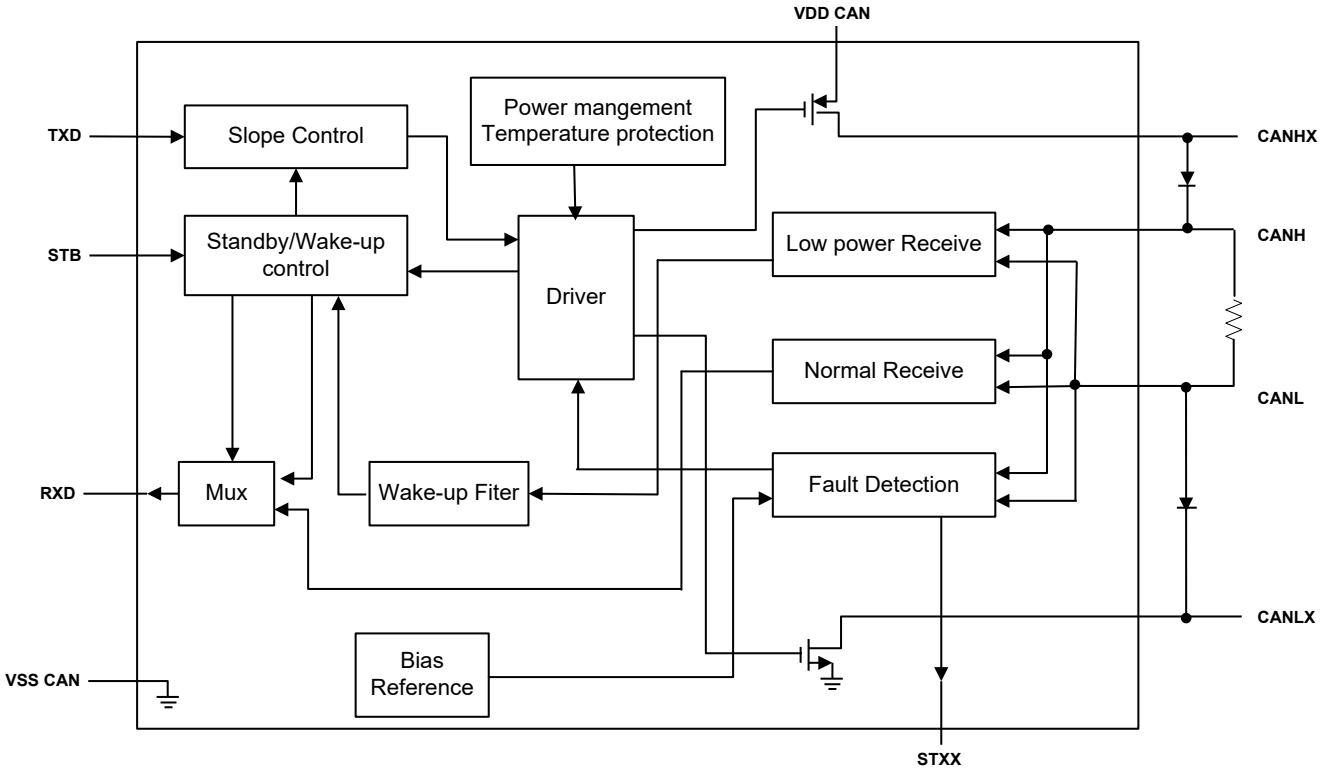
## 5.1 General Description

The CAN transceiver meets or exceeds the specifications of the ISO 11898 standard for use in application employing a Controller Area Network (CAN). As CAN transceivers, these devices provide differential transmit and receive capability for a CAN controller as signaling rates of up to 1megabit per second(Mbps).

## 5.2 Features

Fully compatible with the ISO 11898 standard
High speed (up to 1MB/s)
Low current standby mode with bus wake-up
Suitable foe 12V and 24V system
Differential receiver with high common mode range for EMI
Voltage source for stabilizing the recessive bus level if split termination is used.
Slope control to reduce radio frequency interface (RFI)
Short circuit proof to battery and ground
Thermally protected
At least 110 nodes can be connected

## 5.3 Block Diagram




**Function Block of IS31IO8972 CAN Transceiver**

## 5.4 Function Description

### Operating modes

The can transceiver provides two modes of operation which are selectable via pin STB.
See following table for a description of the modes of operation.

| MODE | STB | RXD | |
|---|---|---|---|
| | | LOW | HIGH |
| Normal | Low | Dominate | Recessive |
| Standby | High | Wake-up request detected | No wake-up request detected |

### Normal mode

In this mode, the can transceiver is able to transmit and recessive data via the bus lines CANH and CANL.
See Fig.10-1 for block diagram. The differential receiver converts the analog data on the bus lines into digital data which is output to pin RXD via the multiplexer (MUX). The slope of the output signals on the bus lines is able to adjusted and optimized in a way that lowest EME is guaranteed.

### Standby mode

In this mode, the transmitter and receiver are switched off, and the low-power differential receiver will monitor the bus lines. A High level on pin STB activates the low-power receiver and the wake-up filter, and after $t_{BUS}$ the state of the CAN bus is reflected on pin RXD. In this mode, the bus lines are terminated to ground to reduce the supply current to a minimum. A diode is added in series with the high-side driver of RXD to $V_{CC}$ in the unpowered state. In normal mode, the diode is passed. This diode is not bypassed in standby mode to reduce current consumption.

### Wake-up

In standby mode, the bus lines are monitored via a low-power differential comparator. Once the low-power differential comparator has detected a dominate bus level for more than $t_{BUS}$, pin RXD will be LOW.

### Over-temperature protection

The output drivers are protected against over-temperature conditions. If the virtual junction temperature exceeds the shutdown junction temperature Tj, the output drivers will be disabled until the virtual junction temperature becomes lower than Tj and TXD becomes recessive again.

### TXD dominate time-out function

A “TXD dominate time-out“ timer circuit prevents the bus lines from being driven to a permanent state if pin TXD is forced permanently LOW by a hardware and/or software application failure. The timer is triggered by a negative edge of pin RXD. If the duration of the LOW level on pin TXD exceeds the internal timer value ($t_{dom}$), the transmitter is disabled, driving the lines into a recessive state. The timer is reset by a positive edge on pin TXD. The TXD dominate time-out time $t_{dom}$ defines the minimum possible bit rate of 40KBaud.

**TRANFAULT (0x76) Transceiver Fault Status Register RW (0x80)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | ENFAULT | - | THMAL | STHD | STHS | STLD | STLS | STHL |
| WR | ENFAULT | - | - | CLRSTHD | CLRSTHS | CLRSTLD | CLRSTLS | CLRSTHL |

Transceiver Fault Status Register is the status flag to notice MCU that transceiver found a fault on CAN bus. When ENFAULT=1 and any fault is issued, the transceiver will be disabled autonomously until all faults are banished from CAN bus and the TRANFAULT[4-0] are cleared by software.

ENFAULT Fault Detector Enable
ENFAULT=1: Enable the fault detection function
ENFAULT=0: Disable the fault detection function

| | |
|---|---|
| THMAL | Transceiver Thermal Protection Status<br>This bit is read only. The status change of the THMAL bit will cause the THMIF interrupt flag.<br>THMAL=1: Transceiver is disabled by the internal thermal shut-down protection<br>THMAL=0: Transceiver is working on normal status |
| STHD | CANH Short to Power<br>STHD=1: CANH short to power for the toleration time defined by CANFAULTTIM<br>STHD=0: CANH is working on normal mode without short to power issue.<br>CLRSTHD=1: Clear STHD and reset the internal toleration time counter.<br>CLRSTHD=0: No effect. |
| STHS | CANH Short to Ground<br>STHS=1: CANH short to ground.<br>STHS=0: CANH is working on normal mode without short to ground issue.<br>CLRSTHS=1: Clear STHS.<br>CLRSTHS=0: No effect. |
| STLD | CANL Short to Power<br>STLD=1: CANL short to power.<br>STLD=0: CANL is working on normal mode without short to power issue.<br>CLRSTLD=1: Clear STLD.<br>CLRSTLD=0: No effect. |
| STLS | CANL Short to Ground<br>STLS=1: CANH short to ground for the toleration time defined by CANFAULTTIM.<br>STLS=0: CANH is working on normal mode without short to ground issue.<br>CLRSTLS=1: Clear STLS and reset the internal toleration time counter.<br>CLRSTLS=0: No effect. |
| STHL | CANH Short to CAHL<br>STHL=1: CANH short to CANL.<br>STHL=0: CANH and CANL are working on normal mode without short together issue.<br>CLRSTHL=1: Clear STHL.<br>CLRSTHL=0: No effect. |

**CANFAULTTIM (0x7D) CAN Fault Detect Time Register RW (0xFF)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | FDTIME[7-0] | | | | | | | |
| WR | FDTIME[7-0] | | | | | | | |

CAN Fault Detect Time Register specifies the toleration time for ‘CANH short to Power’ and ‘CAHL short to Ground’ faults. This register is also protected by CANPTLOCK. Before program CANFAULTTIM, the CANPTCLOCK must be unlocked by writing ‘0xA6’. The toleration time is ‘FDTIME[7-0] + 1’ multiplied by the bit time. In the TRANFAULT register, the STHD flag will be active when CANH short to power over the toleration time, and the STLS flag will be active when CANL short to ground over this toleration time.

# 6. ANALOG PERIPHERAL

## 6.1 On-Chip 1.8V Regulator

The core logic is supplied by the 1.8V output from the on-chip regulator that regulates VDD (3.0V to 5.5V). There is variation of this 1.8V due to chip to chip difference. Because this 1.8V is also used to generate the reference for IOSC, the relative accuracy of this supply voltage is important.

**REGCNF (0x77) Regulator Configuration Register RW (0x0F)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RD | IBRXTRM[2-0] | | | REGTRM[2-0] | | | HYSEN | TMPSEL |
| WR | IBRXTRM[2-0] | | | REGTRM[2-0] | | | HYSEN | TMPSEL |

Regulator Configuration Register (REGCNF) is protected by "Protection Locker Register". MUC must unlock the security bit by programming an “A6H” into Protection Locker Register before program the register.
REGCNF is used to adjust four parameters includes thermal protection, Rx’s hysteresis, output of regulator and the bias current of Rx's comparator. The following table shows detail content of circuit application. The default value of REGCNF is 0x0F. However, to get optimal performance, we strongly recommend users to write the value 0x93 to REGCNF as its initial value. This should be done prior to trim of IOSC or other analog peripherals.

IBRXTRM Bias Current Selector of the receiver comparator

| IBRXTRM[2] | IBRXTRM[1] | IBRXTRM[0] | IB_RX (uA) |
|---|---|---|---|
| 0 | 0 | 0 | 27.5 (default) |
| 0 | 0 | 1 | 25 |
| 0 | 1 | 0 | 22.5 |
| 0 | 1 | 1 | 20 (recommend) |
| 1 | 0 | 0 | 17.5 |
| 1 | 0 | 1 | 15 |
| 1 | 1 | 0 | 12.5 |
| 1 | 1 | 1 | 10 |

REGTRM Regulator Voltage Trimmer

The content of this register trims the 1.8V regulator output level. After reset, REGTRM[2-0] is loaded with default 011 to set the regulator to its mean. MCU should access this information and set the calibrated value into REGTRM[2-0] to get the 1.8V on PAD VDD18. This should be done prior to trimming of IOSC or other analog peripherals. The following table lists the REGATRM value versus the VDD18 value. Note the measurement need to wait for a certain time after setting to allow the regulator to be stable. The time required depends on the decoupling capacitor on PAD VDD18. Typical delay should be around 10msec.

| REGTRM[2] | REGTRM[1] | REGTRM[0] | VDD18 (V) |
|---|---|---|---|
| 0 | 0 | 0 | 1.65 |
| 0 | 0 | 1 | 1.68 |
| 0 | 1 | 0 | 1.71 |
| 0 | 1 | 1 | 1.75 (default) |
| 1 | 0 | 0 | 1.8 (recommend) |
| 1 | 0 | 1 | 1.85 |
| 1 | 1 | 0 | 1.92 |
| 1 | 1 | 1 | 2.01 |

HYSEN Hysteresis Voltage Enable of the receiver comparator

HYSEN=0: Enable hysteresis voltage function and increase 19mV range of the receiver.

HYSEN=1: Disable hysteresis voltage function.

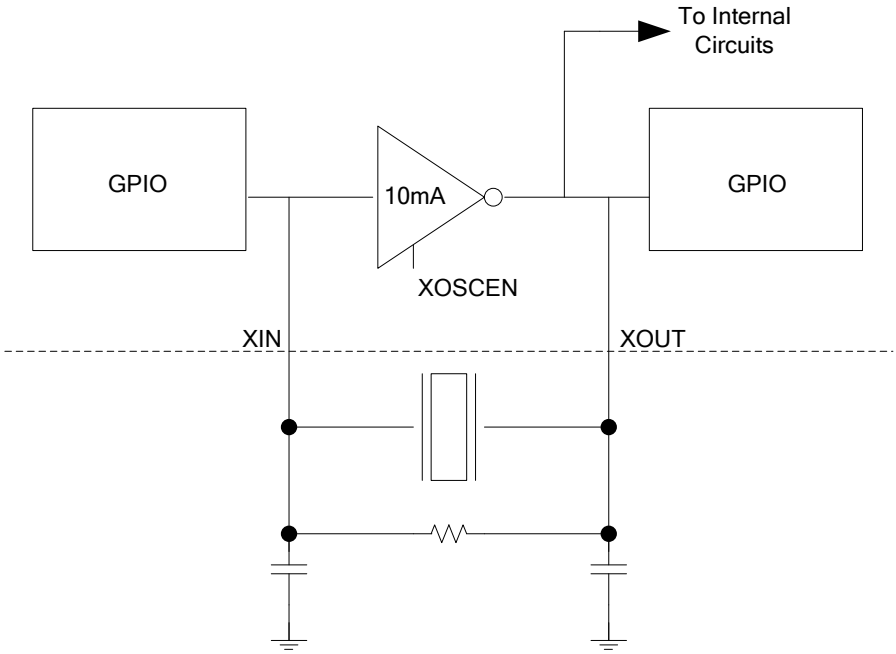
TMPSEL Thermal Protection Selector

TMPSEL=0: The thermal protection is defined around 120℃.

TMPSEL=1: The thermal protection is defined around 175℃.

## 6.2 Crystal Oscillator (XOSC)

Crystal oscillator (XOSC) provides a very accurate clock source for the system. The default for XOSC is in disabled state after power on or reset. Please note when enabling XOSC from disabled state to enabled state, it usually takes 10s of mili-second for XOSC to stabilize. The software needs to take this into considerations before switching into XOSC clock. Typically, the crystal pins are shared with GPIO pins. The software also should take care that the shared GPIO function is properly turned off not affecting the crystal oscillation.
The XOSC circuit is described in the following block diagram.




An external feedback resistor typically ranging from 1M Ohm to 4M Ohm is required. The capability of a stable oscillation depends on several factors, the operating supply voltage, the frequency of crystal, the intended operating temperature range, the quality of the crystal, and the external capacitance load.

| XOSCEN | XOSC Operations |
|---|---|
| 0 | Powered down. |
| 1 | High power for 5V up to 24MHz. |

## 6.3 Internal Oscillator

The internal oscillator is peripheral as it provides the default clock source after reset and other critical timing. The internal oscillator has the salient features that it behaves well during the enable and disable transient. No clock glitches or extra clock edge is generated during the on/off transition, and the oscillator can reach to stable oscillations within very short time typically within 10 cycles. The IOSC consumes around 200uA when enabled. The IOSC is always enabled except entering STOP mode or clear IOSCEN to zero.

| IOSCEN | IOSC Operations |
|---|---|
| 0 | Powered down. |
| 1 | Enable IOSC, 16MHz clock output |

## 7. TEST MODES

For engineer debugging and mass production requirement, IS31IO8972 supports four test modes, CAN Self-Loop Test Mode, CAN Transceiver Only Mode, Automatic Test Pattern Generation (ATPG) Mode and Memory Built-in Test (MBIST) Mode. CAN Self-Loop Test Mode is described in the last section of “CAN Controller” chapter.
Automatic Test Pattern Generation (ATPG) Mode is active by tying “TMODE” input pin to high. Tying “MSEL0” input pin to low and “SPI_SI/MSEL1” to high will enable CAN Transceiver Only Mode.

| TMODE | MSEL0 | SPI_SI/MSEL1 | Function |
|---|---|---|---|
| 1 | X | X | Automatic Test Pattern Generation (ATPG) Mode |
| 0 | 1 | X | Normal mode and SPI interface is selected |
| 0 | 0 | 0 | Normal mode and $I^2C$ interface is selected |
| 0 | 0 | 1 | CAN Transceiver Only Mode |

Test Mode Selection by Input Pins

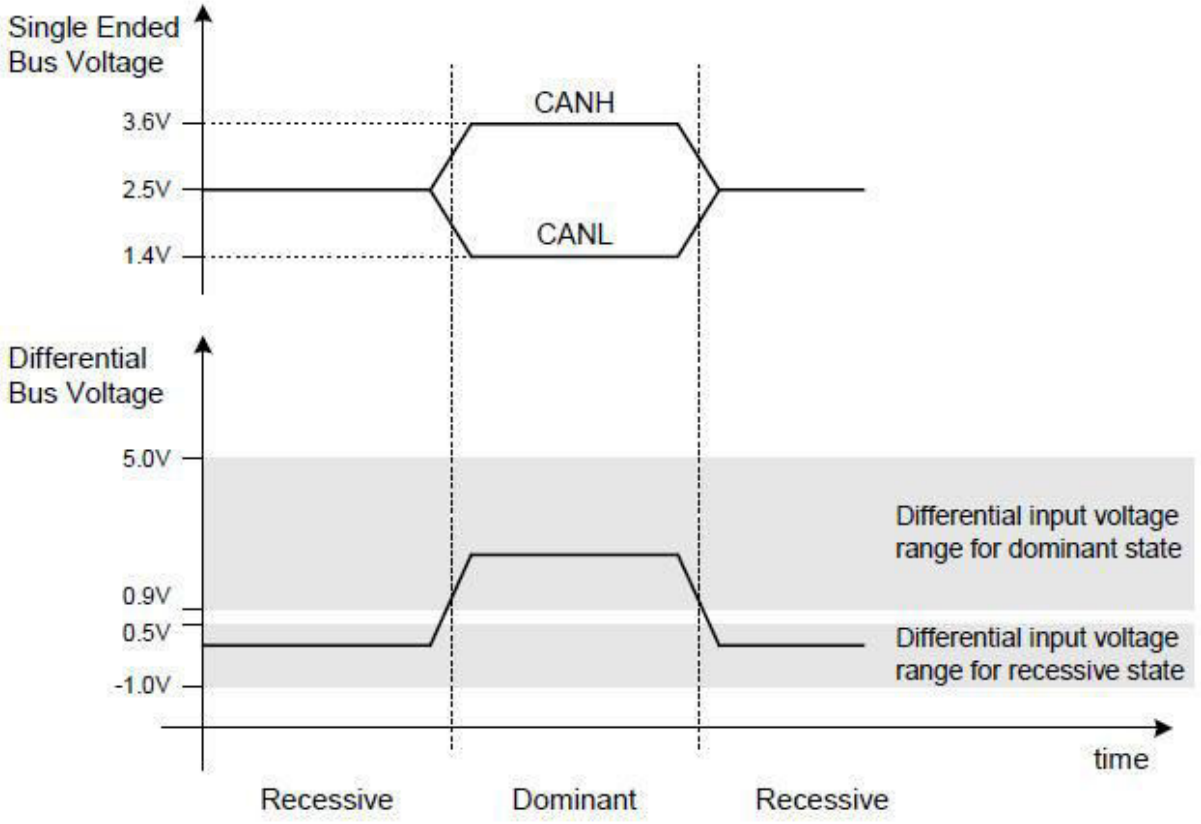
Under CAN Transceiver Only Mode, the internal signal ‘CAN_TX’, ‘CAN_RX’, and ‘CAN_STB’ will be re-directed to IO PAD ‘SPI_CK/SCL’, ‘SPI_SO/SDA’, and ‘SPI_CSn/GPIO0’. User can verify the embedded transceiver by these test pads.
Set the bit-2 of CAN Test Enable Register at address 0x7F after unlock the CAN Test Locker register by programming an “A6h” will enable the Memory Built-in Test (MBIST) mode. The built-in test circuit will verify the embedded SRAM and all normal function will be disabled under the MBIST mode. User can check the “MBIST Done” and the “MBIST Fail” signal on GPIO0 and GPIO1 pads.

## 8. Electrical Characteristics

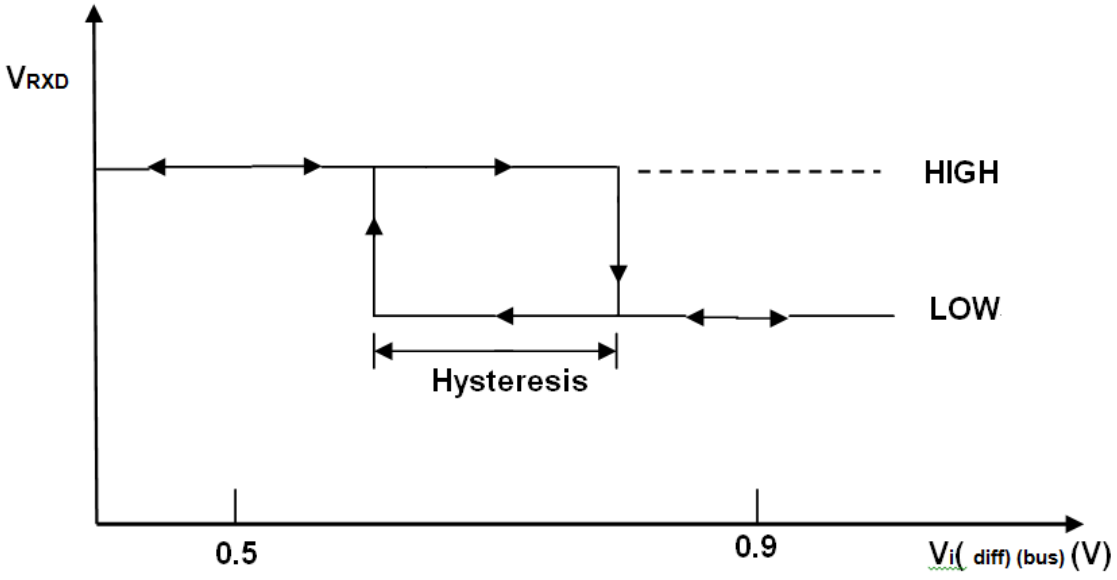
| SYMBOL | PARAMETER | CONDITIONS | MIN. | TYP. | MAX. | UNIT |
|---|---|---|---|---|---|---|
| **Power Supply** | | | | | | |
| VDDIO | Supply voltage | Standby or normal mode | 2.5 | 5 | 5.5 | V |
| VDD_CAN | CAN Bus supply voltage | Standby or normal mode | | 5 | | V |
| Icc | Supply current | Standby mode | 250 | 500 | 750 | uA |
| | | Normal mode<br>Recessive, Vtxd=Vcc<br>Dominate, Vtxd=0V | <br>2.5<br>30 | <br>5<br>50 | <br>10<br>70 | mA |
| **Bus lines (pins CANH and CANL)** | | | | | | |
| Vo (dom) | Dominant output voltage | Vtxd = 0V<br>Pin CANH<br>Pin CANL | <br>2.75<br>0.5 | <br>3.5<br>1.5 | <br>4.5<br>2.25 | <br>V<br>V |
| Vdiff | Differential bus output voltage (Vcanh– Vcanl) | Vtxd = 0; dominate;<br>RL = 60Ω | 1.5 | - | 3.0 | V |
| | | Vtxd = Vcc;<br>recessive; no load | -12 | - | 12 | mV |
| Vo (reces) | Recessive voltage | Normal mode;<br>Vtxd = Vcc; no load | 2 | 0.5Vcc | 3 | V |
| Io (sc) | Short-circuit output current | Vtxd = 0V<br>Vcanh = -18V<br>Vcanl= 25V | <br>-40<br>40 | <br>-<br>- | <br>-160<br>160 | <br>mA<br>mA |
| Io (reces) | Recessive output current | -18V < Vcan < +25V | -2.5 | - | 2.5 | mA |
| Vdiff (th) | Differential receiver threshold voltage | -12V< Vcanl <+12V<br>-12V< Vcanh <+12V<br>Normal mode<br>Standby mode | <br><br>0.5<br>0.4 | <br><br>0.7<br>0.7 | <br><br>0.9<br>1.15 | <br><br>V<br>V |
| Vdiff (th) | Differential receiver threshold voltage (deviation) | Normal mode<br>-12V< Vcanl <+12V<br>-12V< Vcanh <+12V | 50 | 70 | 100 | mV |
| Ri (cm) | Common-mode input resistance | Standby or normal mode | | 300 | | KΩ |
| Ri (dif) | Differential input resistance | Standby or normal mode | | 600 | | KΩ |
| Ci (cm) | Common-mode input capacitance | Vtxd = Vcc; not tested | - | - | 20 | pF |
| Ci (dif) | Differential input capacitance | Vtxd = Vcc; not tested | - | - | 10 | pF |
| **Timing characteristics** | | | | | | |
| tBus | Dominate time for Wake-up via bus | Standby mode | 0.75 | 2.5 | 5 | |
| td (stb-norm) | Delay standby mode to normal mode | Normal mode | 5 | 7.5 | 10 | us |
| **Temperature Characteristics** | | | | | | |

| SYMBOL | PARAMETER | CONDITIONS | MIN. | TYP. | MAX. | UNIT |
|---|---|---|---|---|---|---|
| Tj | Shutdown junction temperature | Thermal shutdown | | 165 | | °C |
| TA | Ambient operating temperature | Standby or normal mode | -40 | | 125 | °C |
| TSTG | Storage temperature | | -65 | | 150 | °C |

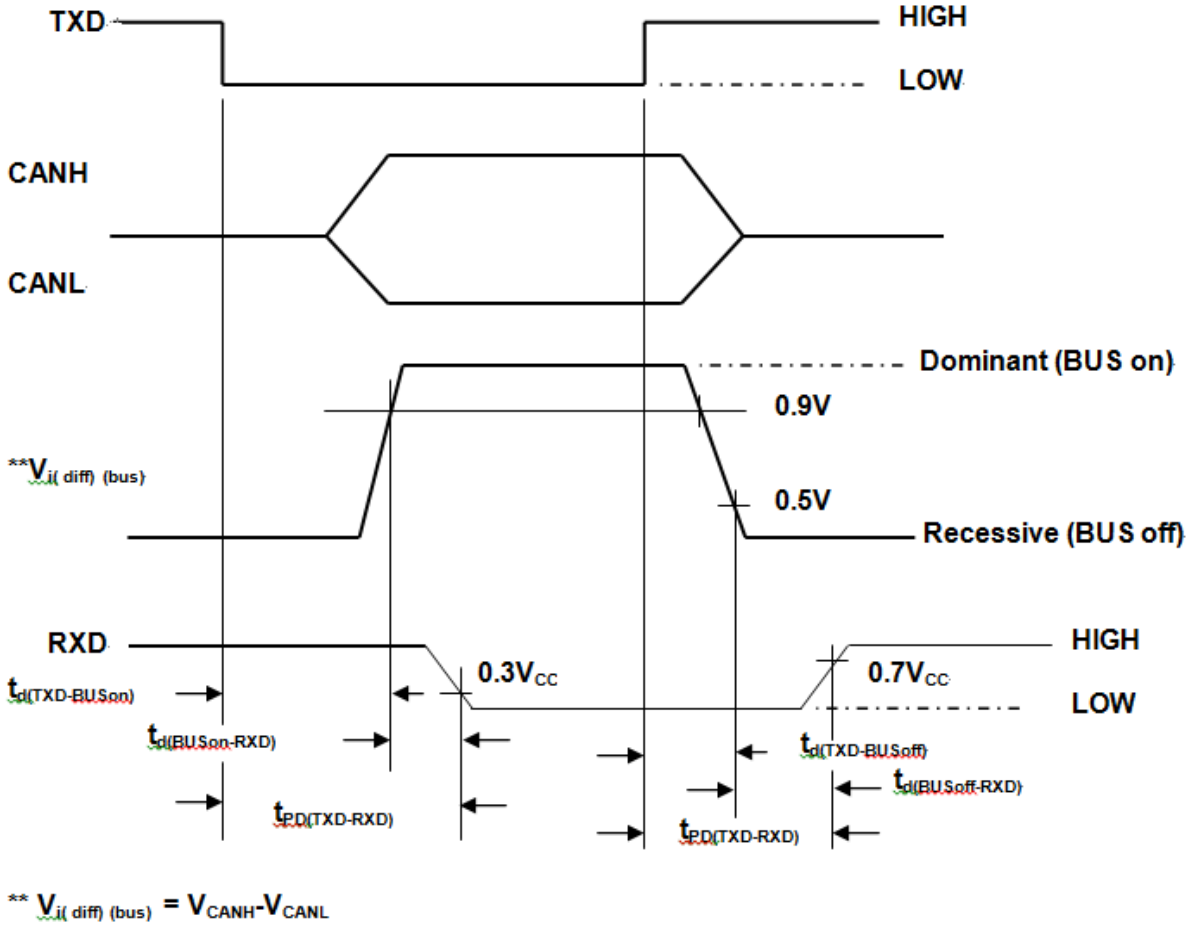
## 8.1 Application information




**Normal bus levels according to ISO 11898**




**Hysteresis of the receiver**

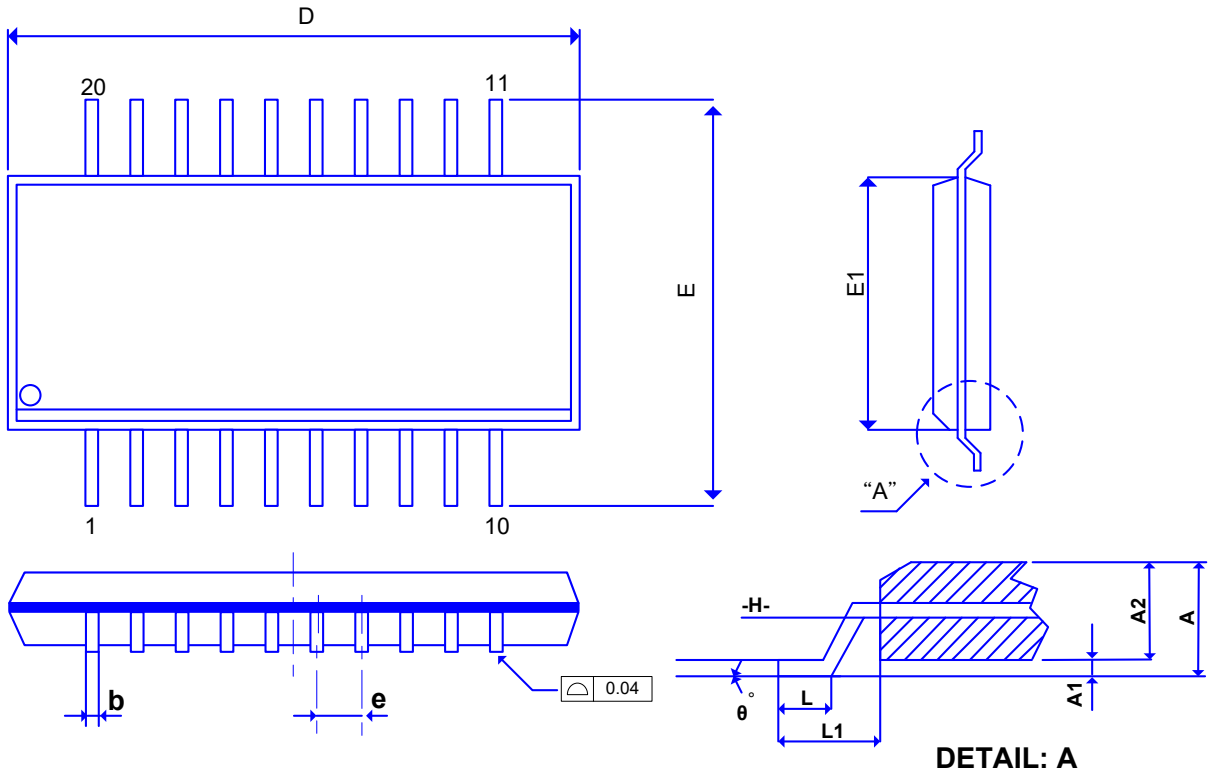



** $V_{i(diff)(bus)} = V_{CANH} - V_{CANL}$

**CAN Transceiver timing diagram**

## PACKAGE OUTLINE

### 20-pin SSOP




| Symbol | Dimensions in Inch | | |
|---|---|---|---|
| | MIN. | NOM. | MAX. |
| A | 0.053 | 0.064 | 0.069 |
| A1 | 0.004 | 0.006 | 0.010 |
| A2 | - | - | 0.059 |
| b | 0.008 | - | 0.012 |
| C | 0.007 | - | 0.010 |
| D | 0.337 | 0.341 | 0.344 |
| E | 0.228 | 0.236 | 0.244 |
| E1 | 0.150 | 0.154 | 0.157 |
| e | 0.025 BASIC | | |
| L | 0.016 | 0.025 | 0.050 |
| L1 | 0.041 BASIC | | |
| $\theta^\circ$ | 0° | - | 8° |

*Notes:*
*JEDEC Outline: MO-137 AD*
*Dimension D does not include mold protrusions or gate burrs. Mold protrusions and gate burrs shall not exceed 0.006" per side. Dimension E1 does not include interlead mold protrusions. Interlead mold protrusions shall not exceed 0.010" per side.*
*Dimension b does note include dambar protrusion/intrusion. Allowable dambar protrusion shall be 0.004" total in excess of b dimension at maximum material condition. Dambar intrusion shall not reduce dimension b by more than 0.[x]2" at least.*

## ORDERING INFORMATION

### Operating temperature -40°C to 125°C

| Order Part No. | Package | QTY |
|---|---|---|
| IS31IO8972-SALS4 | SSOP-20, Lead-free | 75/Tube |

## Revision History

### V0.94

1. Revise Block Diagram.
2. Move “Electrical Characteristics” to the last chapter.
3. Change the max. operating temperature range from 85°C to 125°C.
4. Remove the internal timing characteristics of TXD and RXD.
5. Revise REGCNF for I2 and I3 version.

### V0.95

1. Chang the part no. name as IS31IO8972.
2. Update ordering information to show the right part no.

### Revision 0A

1. Change to ISSI preliminary release version number.

### Revision 0B

1. Update Io (sc) and Io (reces) electrical characteristics.