MATES - Modular Automatic Test Equipment System
User's manual

Viresco μCS / Micro Control Systems

October 17, 2019

Contents

I. General description

1 Acronyms and glossary

Acronyms

ADC : Analog to Digital Converter.

BIT : Built - In Test.

CAN : Controller Area Network.

DAC : Digital to Analog Converter.

DC : Direct Current.

IDC : Insulation Displacement Connector.

LE : Little Endian - least significant byte has the lowest address in memory.

MATES : Modular Automatic Test Equipment System.

PSU : Power Supply Unit.

RAZ : Read As Zero.

SCBIT : Single and Continuous Built - In Test.

SCPI : Standard Commands for Programmable Instruments.

VISA : Virtual Instrument Software Architecture.

VTB : Verocel Test Box.

Glossary

black-box testing : Refers to a situation when only input/output signals are manipulated from outside of the device and the device itself runs its firmware uninterrupted.

CSMA/CD : The CSMA stands for Carrier Sense Multiple Access. What this means is that every node on the network must monitor the bus for a period of no activity before trying to send a message on the bus (Carrier Sense). Also, once this period of no activity occurs, every node on the bus has an equal opportunity to transmit a message (Multiple Access). The CD stands for Collision Detection. If two nodes on the network start transmitting at the same time, the nodes will detect the "collision" and take the appropriate action. For CAN bus, the collision is solved by the bit arbitration.

ESD : Electrostatic discharge; the sudden and momentary electric current that flows between two objects at different electrical potentials.

TH : Test Harness. The Test Harness is the collective name for the components that make up the testing environment.

2 Overview

Each MATES node is designed to serve a simple function such as Digital to Analog Converter (DAC) or Analog to Digital Converter (ADC) device. To create a complex test system, different types of MATES nodes are combined into a single setup that serves multiple functions.

A node has its own low voltage power supply, a Controller Area Network (CAN) bus interface and RS-232 interface. Node can be used as a standalone device or can be connected (using CAN) with other nodes to form a complex system. In the latter case a designated node serves as communication bridge between the MATES system and the PC controlling it.

As much as 10 devices of the same type can be present in the same system (see Setting the CAN node address).

For MATES generation 1 (MK1), this means that a single test system can contain as much as 400 DAC channels, 400 ADC channels, 200 3.3 V inputs, 200 3.3 V outputs, 200 5 to 30 V inputs and 200 5 to 30 V outputs.

As mentioned, the devices are connected together using robust, industrial grade CAN bus. CAN is multimaster, CSMA/CD bus that uses simple cabling and can connect as much as 127 devices (when running CANopen protocol).

The MATES system is modular. The modularity is based on the following node properties:

- each device is completely independent from other devices;
- each device has its own low voltage power supply making it less susceptible to interference from other nodes;
- the CAN interface as well as the power supply sources are electrically isolated making them independent in terms of ground connection - each device can be used with different ground potential or can even be located in different building using different main PSU;
- a node can be added / deleted / replaced without even turning off the system power.

II. MATES nodes - common information

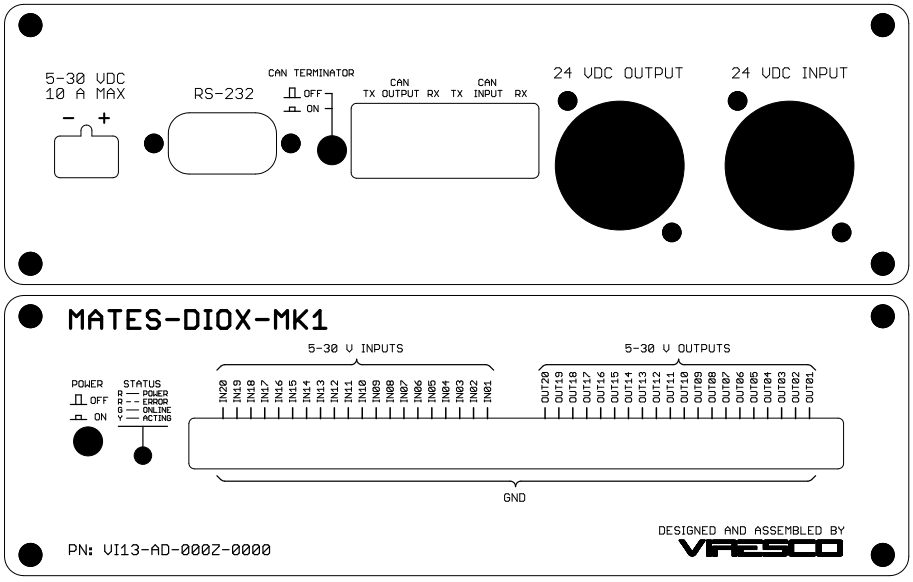
This part contains information common for all types of MATES nodes.

1 Hardware overview

All devices have the same form factor: 165 x 52 x 220 mm (W x H x L). Rear side contains foreign voltage supply connector (MATES-DIOX-MK1 only), RS-232 connector, CAN terminator switch, CAN bus connectors and power supply connectors.

The front side is occupied by the power button, status LED and two 2x20 IDC male connectors for the test signals.

Figure 1: Rear and front panel of MATES-DIOX-MK1



1.1 Power button

The power button is used to connect the external power supply (`24 VDC INPUT`) to the device internal circuitry as well as to the output power connector (`24 VDC OUTPUT`) - if enabled (see 2.3). This way one node can draw its power from second node so only one power supply cable is needed to provide power to all devices in the system.

1.2 Status LED

On the front panel, a single bi-colour LED is used to provide status information. The device state is signalled as follows:

- RED colour denotes the power was applied to the device but the device is not operational yet.
- blinking RED colour means that the device is booting (on start-up) or an error condition occurred.
- GREEN colour shows that the device is operational - this is the normal state after device's successful start-up.
- YELLOW colour means that the device is busy executing commands - it is acting.

Additionally, when device firmware is being updated, the RED and YELLOW colours are shown in turns.

1.3 Signal connectors

Every node features two 2x20 Insulation Displacement Connector (IDC) male sockets. These sockets use standard 2.54 mm (0.1 in) pin spacing. Both sockets have a slot that mates with the plug key to ensure right plug-socket orientation (this feature requires that a standard IDC plug is used).

The connectors have only odd-numbered pins (upper) connected to signals. The even-numbered pins (lower) are all connected to ground to improve noise immunity. Pins #1 are located in upper-right corner of the connectors.

1.4 Foreign voltage connector (MATES-DIOX-MK1 only)

This connector is used to deliver external voltage to power the input - output circuits. It uses standard Phoenix Contact 5.08 mm female plug and male socket.

→ Make sure the external voltage polarity is right as clearly indicated at the rear of the case.

1.5 RS-232 connector

Female DE-9 socket is used for the RS-232 interface. This connection uses only RXD and TXD lines (no hardware handshake). A straight serial cable shall be used to connect to PC.

1.6 CAN terminator switch

This switch is used to turn the 120 Ω CAN bus terminator on / off. See 2.2.

1.7 CAN connectors

The CAN bus uses standard twisted pair cables (straight) and RJ-45 modular connectors. Each MATES node has two connectors to simplify cabling (`CAN INPUT` and `CAN OUTPUT`).

1.8 Power connectors

The power connectors are using Canon standard XLR 3-pin sockets. Male socket is used as the power input (`24 VDC INPUT`) while female socket is used as the power output (`24 VDC OUTPUT`). This arrangement guarantees that the power pins are never exposed. The pin mapping is as follows:

- pin #1 - PSU ground
- pin #2 - PSU +24 VDC
- pin #3 - ESD ground

2 Device configuration

This section describes the activities needed to perform configuration of the MATES devices. Information contained here is common for all devices. Additional device - specific activities are described in particular device's parts of this document (if applicable).

2.1 Setting the CAN node address

Each device has to have unique CAN address. The default addresses for different devices types are given in Default CAN addresses.

Table 1: Default CAN addresses

Device type	CAN address		
	Default	Minimum	Maximum
MATES-DIO3-MK1 MATES-DIO3-MK1.2	20	20	29
MATES-UCC-MK1	30	30	39
MATES-DIOX-MK1	40	40	49
MATES-DAC5-MK1	50	50	59
MATES-ADC5-MK1	60	60	69

If more than one device of the same type is used in the system, the CAN addresses of the second (third, fourth, etc.) device of the same type need to be adjusted in order to be able to access all the devices.

Example

Let's suppose that the system contains three 3.3 V digital IO nodes (MATES-DIO3-MK1). By default, all of them have CAN address of 20. We need to adjust CAN address of the second and third device. It is recommended to use the next addresses as follows:

Device	Address
First MATES-DIO3-MK1	20 (set by default)
Second MATES-DIO3-MK1	21
Third MATES-DIO3-MK1	22

The procedure of setting the CAN address can be conducted using `mates.dll` API (either C, Python or C#) or by using the VISA interface.

In all cases the new address is written to the respective CANID register (CANID (MATES-DIO3-MK1), CANID (MATES-DAC5-MK1), CANID (MATES-ADC5-MK1), CANID (MATES-DIOX-MK1) or CANID (MATES-UCC-MK1)).

Below is an example of C and Python code used to change the CANID value.

Listing 1: Setting CANID in C

```c
/* Open MATES handle. */
MATES_HANDLE h = mates_open("default.mon", 0);

/* Check if node is present. */
if (mates_discover_single_node(h, mates_adc5_mk1_1) == UOS_STATUS_OK)
{
    /* Increase CANID by one. */
    int old_cid;
    (void)mates_get_register(h, mates_adc5_mk1_1, REG_MATES_ADC5_MK1_CANID, &old_cid);
    (void)mates_set_register(h, mates_adc5_mk1_1, REG_MATES_ADC5_MK1_CANID, old_cid
    + 1);
}
mates_close(h);
```

Listing 2: Setting CANID in Python

```python
import mates

# Use Python context manager to open and close the handle.
with mates.Mates("default.mon") as m:
    # Check if node present.
    if m.discover_node(m.mates_dio3_mk1_1):
        # Alter the CAN ID.
        old_cid = m.get_reg(m.mates_dio3_mk1_1, m.regs.REG_MATES_DIO3_MK1_CANID)
        m.set_reg(m.mates_dio3_mk1_1, m.regs.REG_MATES_DIO3_MK1_CANID, old_cid + 1)
```

2.2 Configuration of the presence of the termination resistor

All devices are equipped with CAN bus terminator[1] which can be activated using push button switch at the rear of the case.

→ Note that exactly two devices should have the termination resistor active; the outmost ones.

2.3 Configuration of the power distribution

The devices may operate from a single power source. Each of them has two power connections: input and output (`24 VDC INPUT` and `24 VDC OUTPUT`). You can use single PSU to power the first device, then connect the power output of the first device with the power input of the second device etc.

Normally each device has its own power switch that controls only this particular device. But MATES devices implement a mode when a single switch controls the power of several devices. This mode is enabled by a single toggle switch located at the rear of device PCB (the case need to be open to access it).

Example

Suppose we have 3 MATES devices powered using single Power Supply Unit (PSU). We would like to control the power of all devices using a single power switch (switch on device #1). To configure the system in this way we have to:

1. Connect the PSU to device #1.

[1] See CAN basic topology

2. Connect power from device #1 to device #2 and from device #2 to device #3.
3. Set the power configuration switch in device #1 in position `X2X1`.
4. Set the power configuration switches in remaining devices in position `X2K1X1`.

When system is configured in this way, the device #1 power switch controls the power of itself and all the remaining devices.

3 Startup sequence

After powering up, each MATES node waits for the firmware update command for 3 seconds then performs the image consistency checks and other BIT and enters the default state (this includes the CAN node auto-start). Do not attempt to communicate with the node (via the RS-232 connection) during the powering up sequence. Doing so will prevent application from starting (at least 5 seconds of serial communication silence is needed to allow application to start).

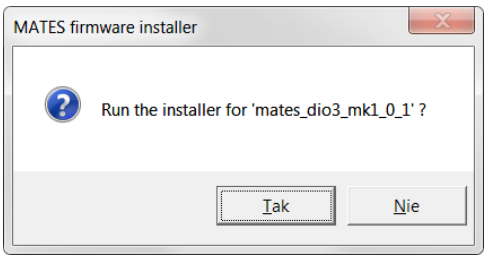
4 Firmware update

Each MATES node is field-upgradeable (see Starting boot loader program). The firmware update uses the serial connection. A single installer file contains all the executables and payload file needed to perform the update.

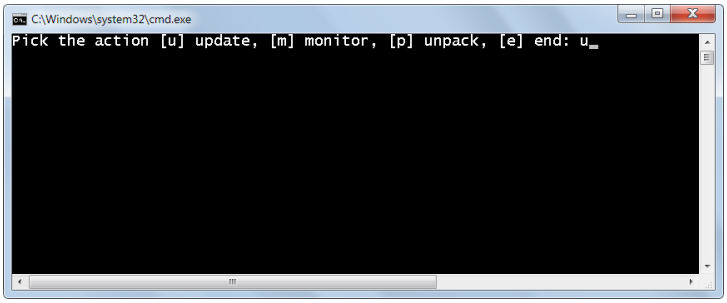
4.1 Direct update of a particular node

To perform an update obtain and execute the respective installer file. The name of the file is in the form: `mates_xxx_y_z.exe` where `xxx` denotes the node name, `y` denotes the major and `z` the minor version of the firmware.

→ The firmware packages can be found in installation directory, see Installation directory content.



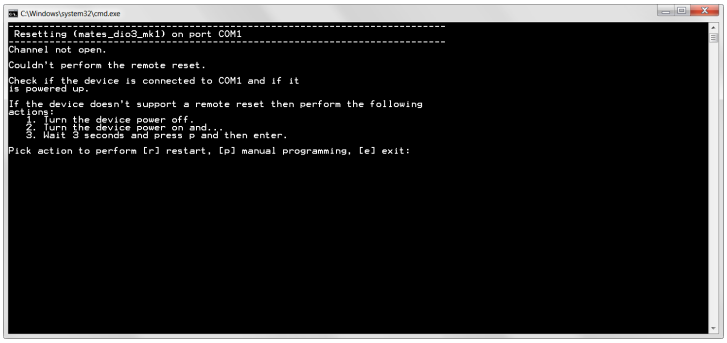
After executing the file the following window is shown:



Selecting "u" and pressing enter starts the update process. Make sure the device is powered up and connected using COM1 serial port. Follow instructions to complete the update.

4.1.1 Recovery procedure in case of firmware update failure

If for some reason the firmware update cannot be completed, the firmware image will be corrupted. To provide a fail-safe scenario, a special power-up sequence is used to be able to start the bootloader. The procedure is described in the console of the same installation program:

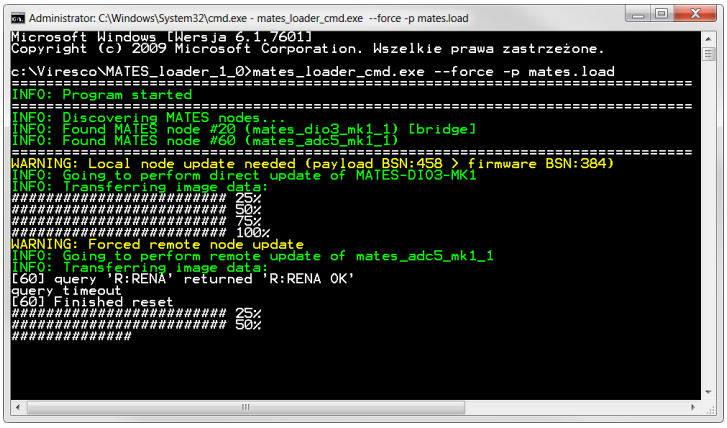


☞ → The fail-save procedure can be attempted multiple times. The application won't start as long as the image is corrupted - this is verified by the payload's CRC.

4.2 Update using MATES loader tool

Besides individual nodes firmware packages, the installation contains a tool designed to be able to update all MATES nodes using single physical connection. This feature is implemented in the `mates_loader_cmd.exe` program.

ℹ → The MATES loader packages containing the `mates_loader_cmd.exe` and a payload file can be found in installation directory, see Installation directory content.



5 Power source

The nominal voltage for the MATES device power source is 24 VDC. Refer to particular devices description for the current consumption ratings.

6 Environmental conditions

- Operational temperature range: 0÷50°C
- Operational relative humidity (non - condensing): 20÷90 %

6.1 Mechanical parameters

Unless otherwise specified, all MATES devices uses the same case form factor.

- Dimensions (WxLxH): 160x225x55 mm
- Weight: approx. 0.5 kg
- Cooling: passive cooling without air-shafts

6.2 Conformity

The device conforms to the following standards:

- PN-EN 61010-1:2011
- PN-EN 61010-031:2005/A1:2009
- PN-EN-61326-1:2009
- 2004/108/WE - EMC

7 Programming

7.1 Using mates.dll

All nodes are accessed uniformly. Programming of a node consists of one or more CAN transactions. Even if the node is accessed directly through the PC interface, it is programmed the same way as any remote node.

There are two basic types of accesses: registers programming and runtime data programming. The former is used to configure a node while the latter is used solely to transmit / receive data. Both accesses are implemented as CAN datagrams which are allocated as PDO services of the CANopen protocol.

The current version of the system (MK1 nodes) doesn't allow executing code directly on a MATES node. Instead, all actions are performed using the two above mechanisms for programming using host (run on PC) software.

All functionality of the MATES system is exposed by a single API implemented in `mates.dll` library - see [3].

7.2 Using VISA queries

MATES nodes implement a subset of VISA / SCPI interface (see [4], [5]). VISA is an acronym for Virtual Instrument Software Architecture. VISA is a Test & Measurement industry standard communication API (Application Programming Interface) for use with test and measurement devices. Some times called a communication driver, VISA allows for the development of programs to be bus independent. Using VISA libraries enables communication for many interfaces such as GPIB, USB, and Ethernet.

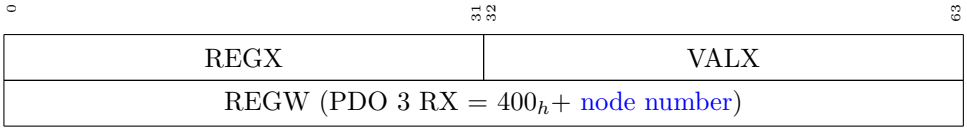
The MATES system uses a simple textual version of the interface (it doesn't implement the actual VISA driver as defined by the IVI standards).

The VISA interface is only available for local node communication. You can use the VISA commands using the `visa_query.py` script. The available commands are listed in List of available VISA queries.

7.3 Using CAN datagrams

Each MATES node can also be programmed using CAN bus. A subset of CANopen datagrams is used to implement entire communication needed to remotely control a node. The following subsections describe CAN datagrams implemented on all MATES nodes. Node - specific datagrams are described in respective "Programming" sections for each node.

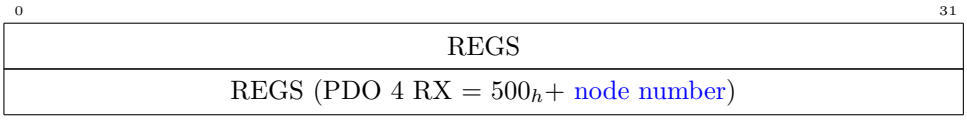
7.3.1 Register setting datagram - REGW



[0:31] REGX — The register number to write (LE).

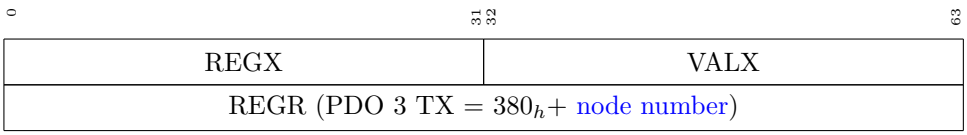
[32:63] VALX — The register value (LE).

7.3.2 Register selector setting datagram - REGS



[0:31] REGS — The register number to select for the subsequent read operations (LE).

7.3.3 Register reading datagram - REGR



[0:31] REGX — The register number being read (LE).

[32:63] VALX — The register value (LE).

7.4 About MATES registers

All MATES registers are 32-bit wide. Each register is a 32-bit signed integer, an IEEE-754 single precision floating point number or a 32-bit wide bitfield. Each register has a unique number by which it is accessed. Registers #0 to #255 are common and are implemented on all MATES nodes.

There are 3 types of register accesses:

- RO - read only access;
- RW - read / write access;
- RWN - read / write access with storing into non-volatile memory.

For RWN, the value written is preserved across power off periods (using EEPROM).

When initial value is given for a register it is understood as:

- for RO and RW registers it means the value just after reset;
- for RWN register it means the factory programmed value.

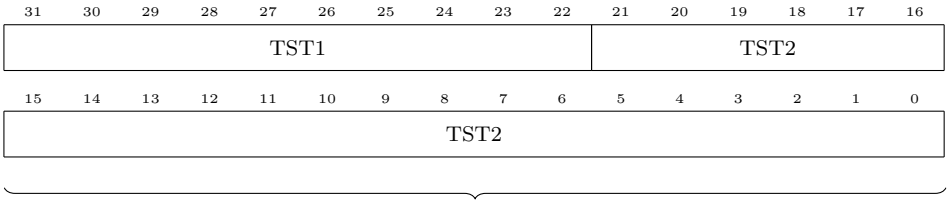
! → Note that for RO registers, the read value may be different than the initial value (in case when the register value is calculated at run time).

When writing register with invalid value (like voltage value in DEF01 which is outside the allowed range), the register writing function returns error and the register value is set using the following rules:

- for integers and floats, if the value written is greater than the maximum, the register is written with the maximum value;
- for integers and floats, if the value written is smaller than the minimum, the register is written with the minimum value;
- for bitfields, all bits are processed individually and similarly limited by the corresponding bits of the minimum / maximum value.

7.5 Common registers

7.5.1 TEST (#0x0) - Used for internal testing, do not use.

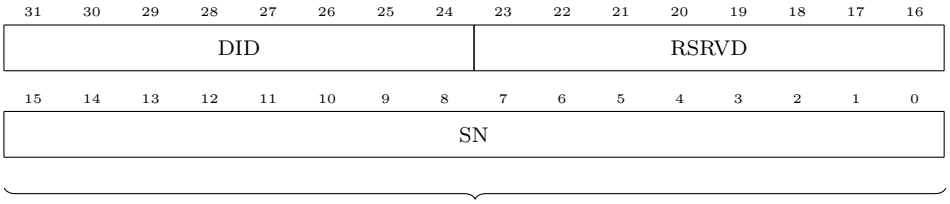


TEST (#0x0, RWN, init.: 0xFF, min.: 0x0, max.: 0xFFFFFFFF)

[31:22] TST1 Field 1.

[21:0] TST2 Field 2.

7.5.2 DIR (#0x1) - Device identification register



DIR (#0x1, RO, init.: 0x0)

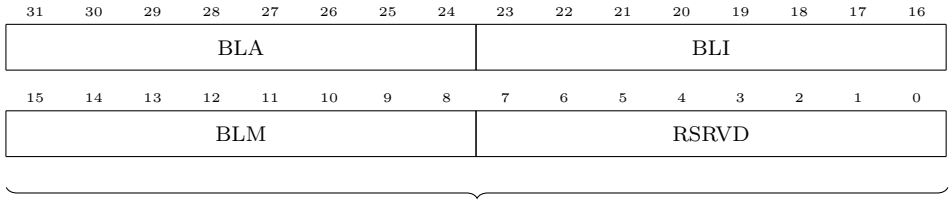
[31:24] DID Device family Identifier:

- 1 for MATES-DIO3-MK1
- 2 for MATES-DAC5-MK1
- 3 for MATES-DIOX-MK1
- 4 for MATES-ADC5-MK1
- 5 for MATES-UCC-MK1

[23:16] RSRVD Reserved for future use - RAZ.

[15:0] SN Device serial number.

7.5.3 BIR (#0x2) - Boot loader identification register



BIR (#0x2, RO, init.: 0x0)

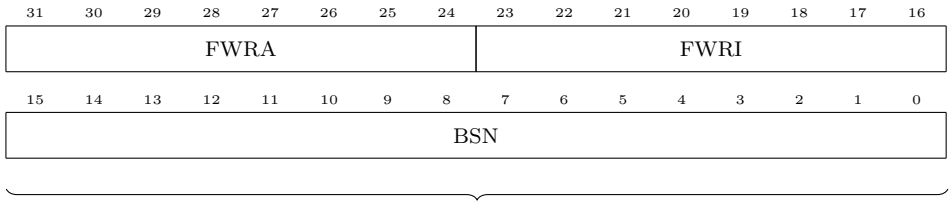
[31:24] BLA — Boot loader major version.

[23:16] BLI — Boot loader minor version.

[15:8] BLM — Boot loader mode.

[7:0] RSRVD — Reserved for future use - RAZ.

7.5.4 FIR (#0x3) - Firmware identification register



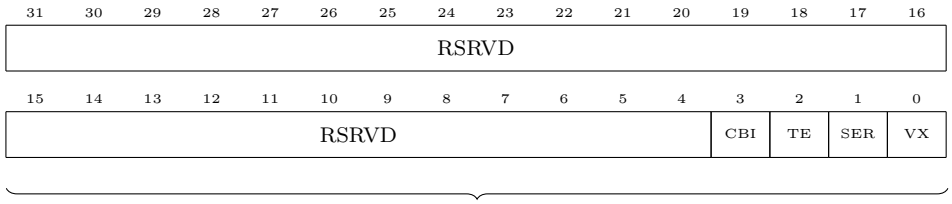
FIR (#0x3, RO, init.: 0x0)

[31:24] FWRA — Firmware major version (production devices have FWRA >= 1).

[23:16] FWRI — Firmware minor version.

[15:0] BSN — Build sequence number.

7.5.5 SR (#0x4) - Status register



SR (#0x4, RO, init.: 0x0)

[31:4] RSRVD — Reserved for future use - RAZ.

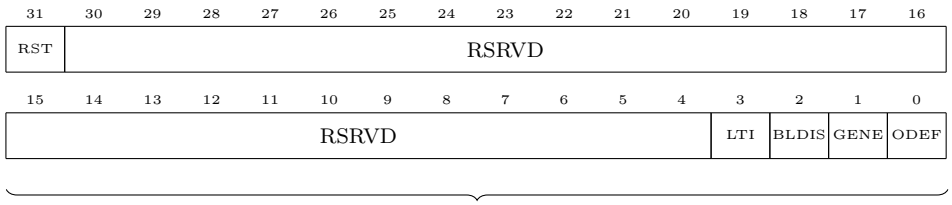
[3] CBI — CAN bridge indicator (1 when device is connected to PC and acts as CAN bridge).

[2] TE — Terminator enable state (1 if CAN terminator is enabled, 0 otherwise).

[1] SER — Software error indicator (1 when any software error occurred, this bit is sticky).

[0] VX — External voltage present indicator (1 when external voltage is present, applicable only for MATES-DIOX-MK1, for other nodes this bit always reads as 0).

7.5.6 CR (#0x5) - Control register



CR (#0x5, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31] RST — Reset request bit. Writing 1 to this bit will cause the device to reset within 1 second. RAZ.

[30:4] RSRVD — Reserved for future use - RAZ.

[3] LTI — Lamp test initiate. Writing 1 to this bit initiates lamp test sequence. The lamp test sequence activates the device's LED indicators using the following procedure:

- turn all LEDs off for one second;
- turn green status LED on for one second;
- turn red status LED on for one second;
- turn CAN RX LED on for one second;
- turn CAN TX LED on for one second;
- restore normal LEDs operation.

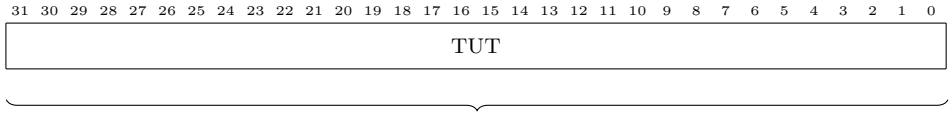
The LTI bit will read as 1 as long as the sequence is in progress. Writing 0 to this bit has no effect.

[2] BLDIS — Boot loader disable. Writing 1 to this bit disables the timed boot loader sequence. The boot loader sequence is re-enabled when device reset is commanded. Writing 0 has no effect. RAZ.

[1] GENE — Generators enable bit, writing 1 to this bit enables outputs generators (applicable only to MATES-DIO3-MK1 and MATES-UCC-MK1, for other devices this bit is reserved and RAZ). You need to clear this bit and then set it if you want to use synchronized generation. See Programmable generators.

[0] ODEF — Set default values for outputs. Writing 1 to this bit causes all outputs to be set to their default values. Writing 0 has no effect. RAZ.

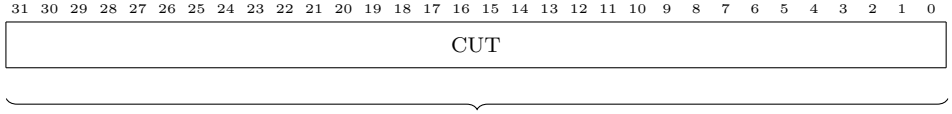
7.5.7 TUT (#0x6) - Total up time register



TUT (#0x6, RO, init.: 0x0)

[31:0] TUT — The device total operating time in seconds.

7.5.8 CUT (#0x7) - Current up time register

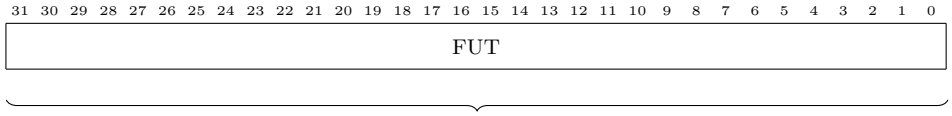


CUT (#0x7, RO, init.: 0x0)

[31:0] CUT — Current operation time (since power up) in seconds.

7.5.9 FUT (#0x8) - Firmware up time register

Device operation time since the last firmware update.

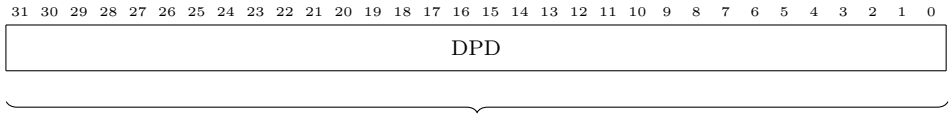


FUT (#0x8, RO, init.: 0x0)

[31:0] FUT — Firmware operating time in seconds.

7.5.10 DPD (#0x9) - Device production date register

Device production date.

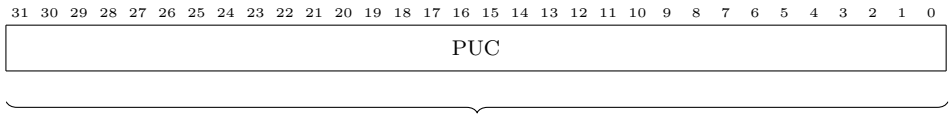


DPD (#0x9, RO, init.: 0x0)

[31:0] DPD — Device production time (UTC) saved as Unix timestamp (compatible to C library time_t).

7.5.11 PUC (#0xA) - Power up counter register

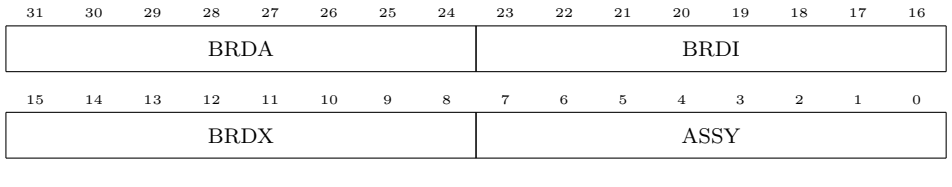
Power up counter.



PUC (#0xA, RO, init.: 0x0)

[31:0] PUC — Number of power up cycles since device production.

7.5.12 HIR (#0xB) - Hardware identification register



HIR (#0xB, RO, init.: 0x0)

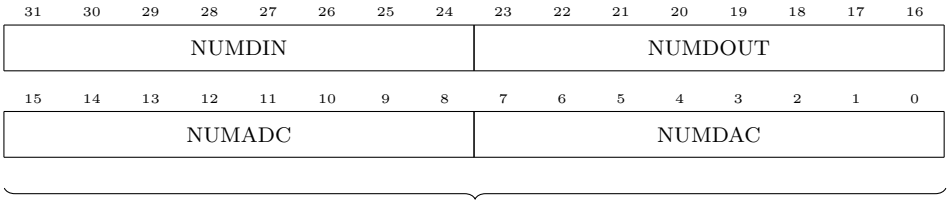
[31:24] BRDA — Board major version. 1 for initial revision.

[23:16] BRDI — Board minor version. This is set to 0 for initial revisions of each major version.

[15:8] BRDX — Board configuration index.

[7:0] ASSY — Assembly variant. 0 for the standard assembly.

7.5.13 CAPS01 (#0xC) - Device capabilities register #1



CAPS01 (#0xC, RO, init.: 0x0)

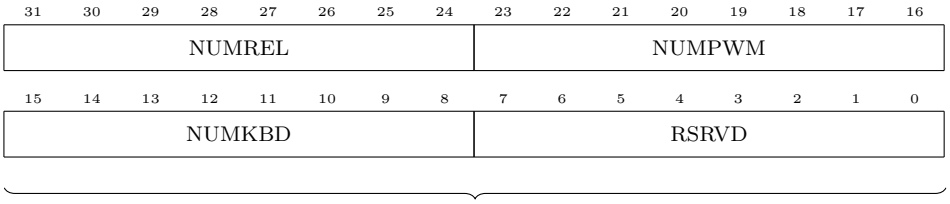
[31:24] NUMDIN Number of digital input channels.

[23:16] NUMDOUT Number of digital output channels.

[15:8] NUMADC Number of ADC channels.

[7:0] NUMDAC Number of DAC channels.

7.5.14 CAPS02 (#0xD) - Device capabilities register #2



CAPS02 (#0xD, RO, init.: 0x0)

[31:24] NUMREL Number of relays.

[23:16] NUMPWM Number of PWM channels.

[15:8] NUMKBD Number of keypad inputs.

[7:0] RSRVD Reserved for future use - RAZ.

8 Programming examples

For a comprehensive set of examples see MATES API documentation [3] (online HTML form) and the content of API reference.

The API documentation contains usage example for every C and Python API function. The sections below are only briefly introducing some of the APIs.

→ The example files quoted in this section can be found in the MATES installation directory, see Installation directory content.

8.1 Direct DLL calls

The MATES system can be used by any language that can access a DLL. The following example is written in C.

Listing 3: mates_test_10.c

```c
#include <stdio.h>
#include "mates.h"

int main(void)
{
    MATES_HANDLE h = mates_open("proxy.mon", 1);
    int success = 0;

    if (mates_discover_single_node(h, mates_dac5_mk1_2) != UOS_STATUS_OK)
    {
        mates_close(h);
        return (0);
    }

    /* Set output to 2.5 V. */
    if (mates_set_dac(h, mates_dac5_mk1_2, MATES_OUT40, 2.5) == UOS_STATUS_OK)
    {
        success++;
        printf("Successfully set DAC voltage value\n");
    }
    else
    {
        mates_print_errors();
    }
    mates_close(h);
    return ((success == 1) ? 0 : -1);
}
```

The example can be compiled into executable file using the following command (MinGW compiler, mates.h and mates.dll are in the same directory as the source file):

```
gcc mates.dll mates_test_10.c -o mates_test_10.exe
```

Consult your C/C++ compiler documentation to find out how to setup library paths and include directory paths to use MATES files from external location (like the MATES installation directory).

8.2 Python wrapper

The Python wrapper is written using Python 2.7 (and will be eventually ported to Python 3.x). Python wrapper is the preferred way of high level MATES programming. The code is very compact and the performance is almost as good as the C interface.

Listing 4: mates_test_18.py

```
import mates


class Dio3(mates.Mates):
    """Represents one MATES-DIO3-MK1 node."""

    def __init__(self):
        super(Dio3, self).__init__("proxy.mon", 1)
        self.node = None
        for node in (self.mates_dio3_mk1_1, self.mates_dio3_mk1_2):
            # Find first DIO3 node accessible on the bus.
            if self.discover_node(node):
                self.node = node
                break

    def get_din(self, channel):
        return super(Dio3, self).get_din(self.node, channel)

    def set_dout(self, channel, value):
        super(Dio3, self).set_dout(self.node, channel, value)

    def __getitem__(self, idx):
        return self.get_din(idx)

    def __setitem__(self, idx, val):
        self.set_dout(idx, val)


# First, make sure the node we are using is connected.
m = mates.Mates("proxy.mon", 1)
if m.discover_node(m.mates_dio3_mk1_1):
    m.close()

    # The with statement below handles opening and closing of
    # the MATES handle. See Mates class constructor for details.
    with Dio3() as d:
        print("First channel: {0}".format(d[0]))
        print("Second channel: {0}".format(d[1]))
        d[2] = 0
        d[3] = 1
        # If we have loopback, then the values below will be
        # the same as the ones set.
        print(d[2])
        print(d[3])
else:
    m.close()
```

9 Tools

This section briefly describes the software tool available for the MATES system. This section is not intended to cover entire functionality of the tools. Refer to the particular tool's manual or command line help for more information.

9.1 Update and calibration tool - `mates_loader_cmd.exe`

This is a command line program used to update firmware of MATES nodes and to perform ADC/DAC calibration where applicable. It uses a single database file (typiccally `mates.load`) which contains:

- firmware images for all types of nodes;

- meta-data used to determine the version of the images;
- non-volatile data for every single node (like e.g. ADC calibration data).

When a firmware update is done, the following procedure is used:

1. Detect (discover) all nodes present in the connected MATES system.
2. Check if update of a node is needed.
3. If the firmware is obsolete, update the image.
4. Upload non-volatile data from the database file into the device.

→ The non-volatile data is always loaded into the nodes, so it will override any existing data. This is done to be able to restore the calibration data in case of a node non-volatile memory failure. If you want to preserve the node's non-volatile data, download it into the database file prior to updating firmware using the `--download` command line option (see 9.1.1).

The `mates_loader_cmd.exe` tool also provides functionality needed to perform ADC or DAC device calibration. This is done in a semi-automatic fashion with little user intervention. The calibration procedure requires presence of a voltmeter of a suitable class and a regulated voltage source.

To use the calibration feature, use the `--calibrate` command line option. The program will perform the calibration of all MATES ADC and DAC nodes discovered. To perform calibration of a chosen device only, turn off the remaining nodes. Also, the program lets you to pick the set of channels to calibrate.

After the calibration of each channel, the calibration data is not only saved in the node's non-volatile memory. It is also written into the database file so you can review the values. The same database file can then be used to update the firmware preserving the calibration settings.

9.1.1 Preserving non-volatile data during firmware update

If your non-volatile data (like calibration data or default output states) differs from the factory programmed values and you want your data to be preserved across firmware update, follow the following procedure:

1. Obtain the update package (`mates_loader_*.exe`).
2. Run the file and choose "unpack".
3. Navigate to the destination directory (displayed during unpacking).
4. Open the command line and enter:

```
1  mates_loader_cmd.exe --download -p mates.load
2  mates_loader_cmd.exe -p mates.load
```

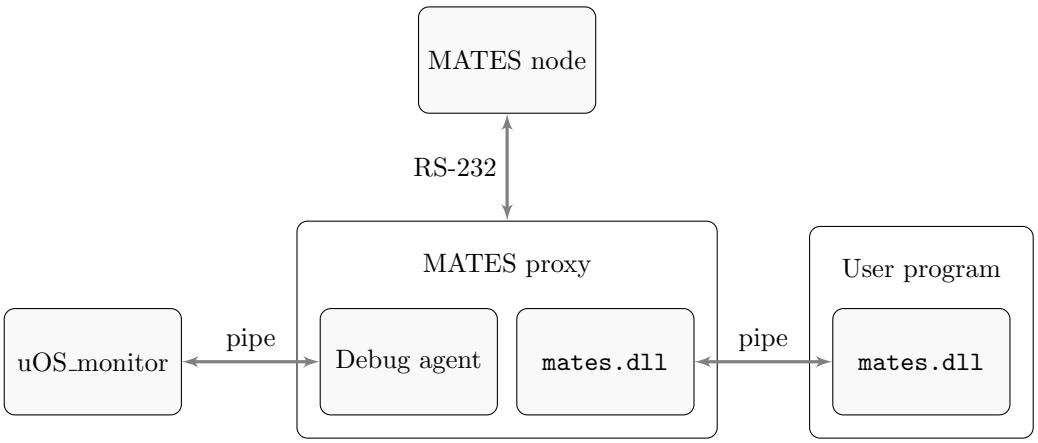
9.2 MATES DLL access proxy server - `mates_proxy.exe`

This tool is designed for debug purposes and can also be used to have remote MATES system access (over the computer network).

The server operates in the following fashion:

The server synchronizes the debug agent and the user program calls. The uOS_monitor program is used to display the MATES data.

Figure 2: MATES proxy operation



III. MATES-DIO3-MK1 — MATES-DIO3-MK1.2 - digital inputs / outputs module

This module is used to interface standard 3.3 V logic signals. It has 20 3.3 V inputs and 20 3.3 V outputs.

The MK1.2+ versions of this device can optionally interface 5 V logic.

1 Properties

1.1 Electrical parameters

The MATES-DIO3-MK1 device provides 20 digital inputs and 20 digital outputs in 3.3 V logic standard.

The MATES-DIO3-MK1.2 device provides 20 digital inputs and 20 digital outputs in 3.3 V or 5 V logic standard.

The inputs of MATES-DIO3-MK1 can be configured using JP5 jumper in three ways (requires opening of the case):

- all inputs are high impedance (jumper removed)
- all inputs have pull-up resistors (jumper in 1-2 position)
- all inputs have pull-down resistors (jumper in 2-3 position)

The factory setting is pull-downs enabled.

The inputs of MATES-DIO3-MK1.2 are configured to use either pull-up resistors or pull-down resistors. This setting is controlled using PSEL (#0x240) - Pull-up / pull-down selection register.

The IO voltage level for both inputs and outputs is set using relay which is controlled by the VIOS field in ODEF (#0x200) - Outputs default value register (MATES-DIO3-MK1.2 only).

Table 2: Device parameters

Parameter	Symbol	Min.	Typ.	Max.
Current consumption[a]	I_N	-	-	60 mA

[a] at V_{IN} = 24 V, no load

2 Programming

2.1 Datagrams

2.1.1 Standard status - HEARTBEAT

7	6	5	4	3	2	1	0
STATUS							
HEARTBEAT (NMT = 700_h+ node number)							

[7:0] STATUS — Communication status of node, according to CANopen.

Table 3: Digital inputs IN01 - IN20

Parameter	Symbol	Min.	Typ.	Max.
$V_{IO} = 3.3\ V$				
Input voltage	V_I	-0.5 V	-	3.8 V
Low level voltage	V_{IL}	-	-	0.8 V
High level voltage	V_{IH}	2 V	-	-
Pull-up / pull-down	R_{PD}	-	10 kΩ	-
$V_{IO} = 5\ V$				
Input voltage	V_I	-0.5 V	-	6.5 V
Low level voltage	V_{IL}	-	-	1.5 V
High level voltage	V_{IH}	3.5 V	-	-
Pull-up / pull-down	R_{PD}	-	10 kΩ	-

Table 4: Digital outputs OUT01 - OUT20

Parameter	Symbol	Min.	Typ.	Max.
$V_{IO} = 3.3\ V$				
Low level voltage	V_L	-	-	0.55 V
High level voltage	V_H	2.5 V	-	-
Output current	I_O	-	-	±10 mA
Transition time	T_t	-	-	10 ns
$V_{IO} = 5\ V$				
Parameter	Symbol	Min.	Typ.	Max.
Low level voltage	V_L	-	-	0.55 V
High level voltage	V_H	3.8 V	-	-
Output current	I_O	-	-	±30 mA
Transition time	T_t	-	-	5 ns

2.1.2 Digital inputs state - DIO3-IN

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
I08	I07	I06	I05	I04	I03	I02	I01	I16	I15	I14	I13	I12	I11	I10	I09	RSVD				I20	I19	I18	I19
byte 0								byte 1								byte 2							
DIO3-IN (PDO 1 TX = 180_h+ node number)																							

[0:15] IXX — The current state of the digital inputs (0 represents low electric state).

[16:19] RSVD — Reserved for future use.

[20:23] IXX — The current state of the digital inputs (0 represents low electric state).

2.1.3 Digital outputs state - DIO3-OUT

This datagram is used by to set the value of the digital outputs of the device. The datagram value can be read back using remote frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
O08	O07	O06	O05	O04	O03	O02	O01	O16	O15	O14	O13	O12	O11	O10	O09	RSVD		MODE		O20	O19	O18	O17
byte 0								byte 1								byte 2							
DIO3-OUT (PDO 1 RX = 200_h+ node number)																							

[0:15] OXX — The current state of the digital outputs (0 represents low electric state).

[16:17] RSVD — Reserved for future use (must be set to zero).

[18:19] MODE — The outputs setting mode:

MODE	Name	Action
00_b	SET	OUT ← O01:O20
01_b	AND	OUT ← OUT & O01:O20
10_b	OR	OUT ← OUT | O01:O20
11_b	XOR	OUT ← OUT ^ O01:O20

! → When reading this datagram, the MODE field is always 0.

[20:23] OXX — The current state of the digital outputs (0 represents low electric state).

2.1.4 Individual digital output state - DIO3-XOUT

This datagram is used to set the value of an individual digital output. The datagram can be read back using remote frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CHANNEL								R1	R2	R3	R4	R5	R6	R7	OUT
byte 0								byte 1							
DIO3-OUTX (PDO 2 RX = 300_h+ node number)															

[0:7] CHANNEL — The channel number to set (0 means OUT01, 19 means OUT20).

[8:14] RX — Reserved for future use (set to 0).

[15] OUT — Output value.

2.2 Programmable generators

The MATES-DIO3-MK1/MATES-DIO3-MK1.2 devices can be programmed to produce simple waveforms on all 20 digital outputs. Each output is programmed using three registers:

- period register (e.g. PRD01);
- output rise time register (e.g. TRISE01);
- output fall time register (e.g. TFALL01).

All registers are programmed using values expressed in microseconds. Setting the period register to 0 µs disables the particular generator. When a generator is disabled, corresponding pin operates as a normal output. The generation for all channels is controlled using GENE bit in CR register. See the following figures for examples of registers settings.

2.2.1 Using regular IO along with generators

You can use regular IO functions when using generators. Channels that have their corresponding period register set to 0 can be controlled using normal procedures.

2.2.2 Manipulating output when generator is enabled

Setting value of an ouptut while the corresponding generator is enabled has no effect but it is not treated as an error. The value set during generator operation is saved in a shadow register and written to output as soon as the period register becomes 0.

2.2.3 Registers access vs the mates_setup_generator() API

There are few differences when programming generators directly using PRDx, TRISEx, TFALLx registers and using the dedicated `mates_setup_generator()` API.

When programming registers directly, no checks are performed and certain values will not produce a valid setup (like e.g. setting TRISEx above PRDx).

On the other hand, the `mates_setup_generator()` provides the additional functionality:

- Checks that all parameters have valid values in terms of generator resolution;
- Checks that period, rise time and fall time have valid values with respect of each other (rise time and fall time are no greater than period);
- If rise time is equal to period, sets the corresponding TRISEx register to 0;
- If fall time is equal to period, sets the corresponding TFALLx register to 0.

2.2.4 Examples

Figure 3: Example generator waveform

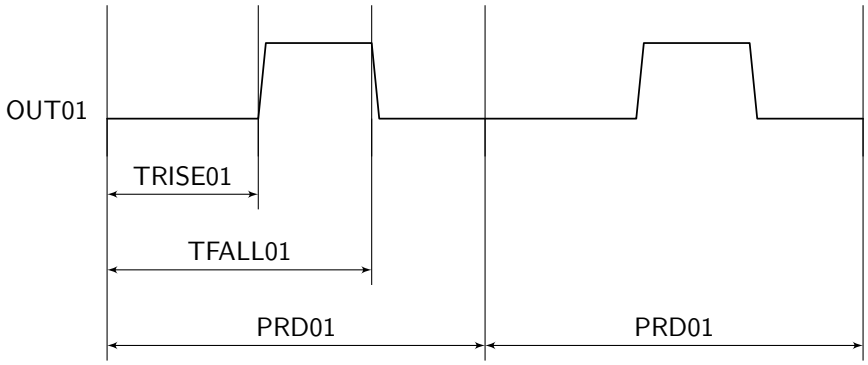
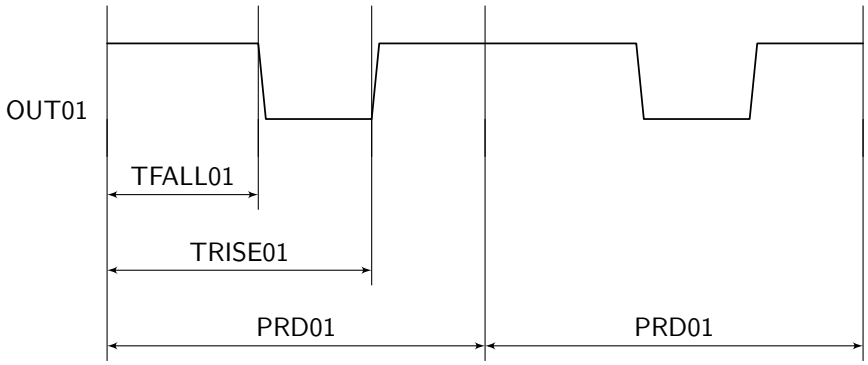


Figure 4: Negative pulse using TFALL < TRISE

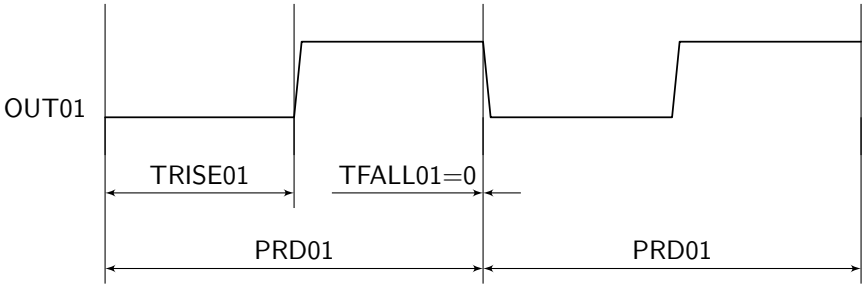


2.2.5 Generator resolution

For MATES-DIO3-MK1/MATES-DIO3-MK1.2, the time resolution is exactly:

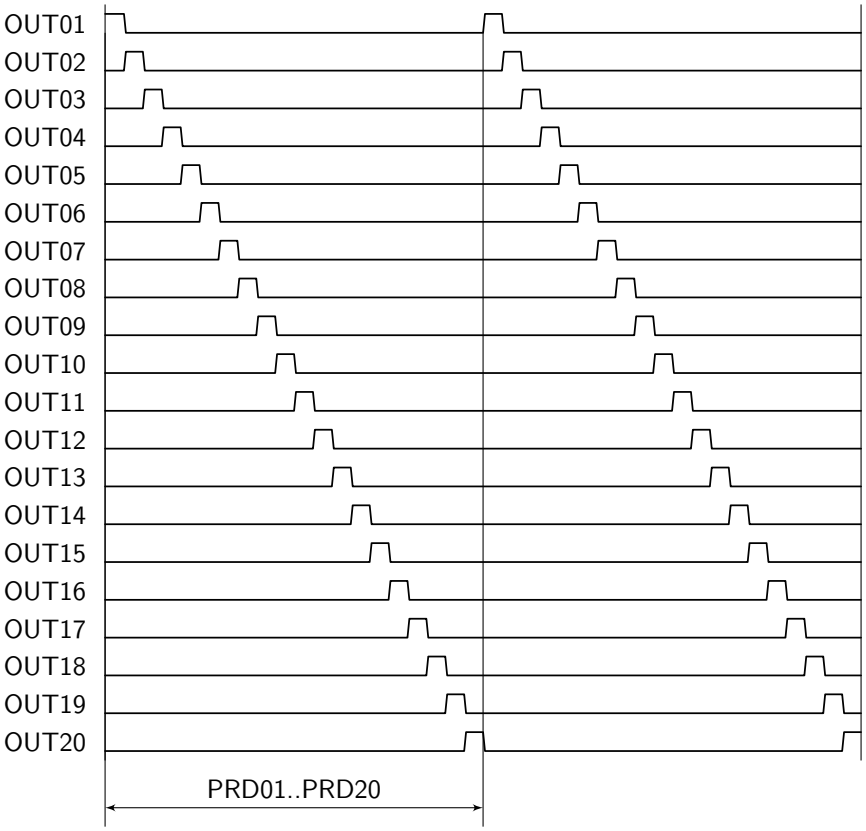
$$T_q = 125\,\mu s \tag{1}$$

Figure 5: Signal fall at period end[a]



[a] Note that you cannot set TRISEx = PRDx or TFALLx = PRDx. These settings will not produce any transitions as the internal clock never reaches value of PRDx (it reaches value of PRDx - 1 and then rolls over to 0). Note however, that the `mates_setup_generator()` has a special handling for this case, see 2.2.3)

Figure 6: Emulating ring counter output



Writing a value that is not a multiple of T_q to any of the three generator registers results in an error. The T_q value may decrease in any future revisions of the hardware (the code backward compatibility will by preserved).

Using the specified T_q value, the highest frequency settings for a square (duty cycle is exactly 50%) waveform can be calculated as shown in table 5.

2.2.6 Using synchronized outputs

If two or more channels are programmed to have exactly the same period (or one period is a multiple of other), the channels run synchronously. You need to program

all channels before starting the generators[2] to use this function. If the generators are not stopped during programming, the respective time bases of generators are not guaranteed to be synchronized (clearing of the GENE bit resets all time bases to 0).

When channels are synchronized, the rise and fall times can be used to create some special patterns like Quadrature encoder waveform.

Figure 7: Synchronized channels

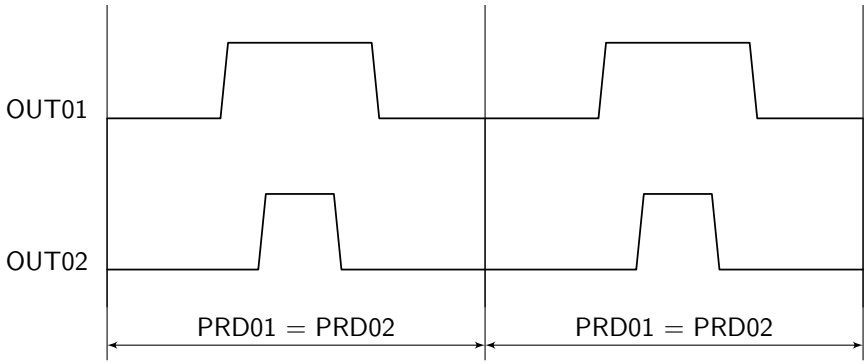
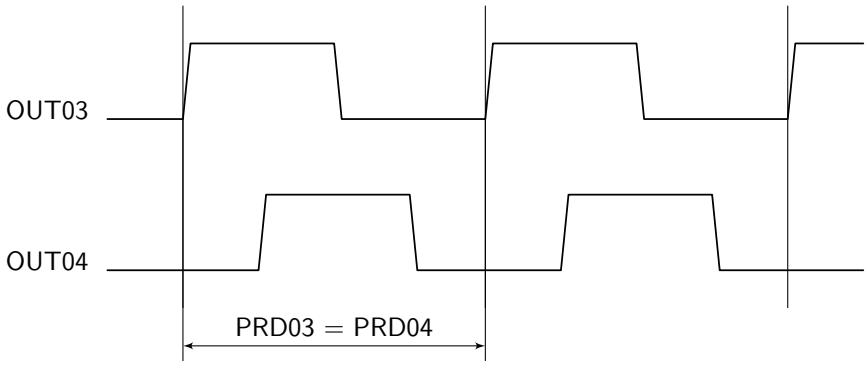


Figure 8: Quadrature encoder waveform



2.2.7 Generator latency

Due to programmatic nature of the signal generation, there is a small latency between actual voltage transitions on different pins even if all channels are programmed to change state at the same time (see OUT01 to OUT20 latency). This parameter, called T_{dly}, changes slightly for different generators configurations. The worst case scenario (all generators enabled, all transitions done in the same time, latency between OUT01 and OUT20) give the latency value of:

$$T_{dly}(max) = 80\,\mu s \tag{2}$$

The best case scenario (delay between two consecutive channels):

$$T_{dly}(min) = 5\,\mu s \tag{3}$$

[2] see GENE bit in CR register

Figure 9: OUT01 to OUT20 latency

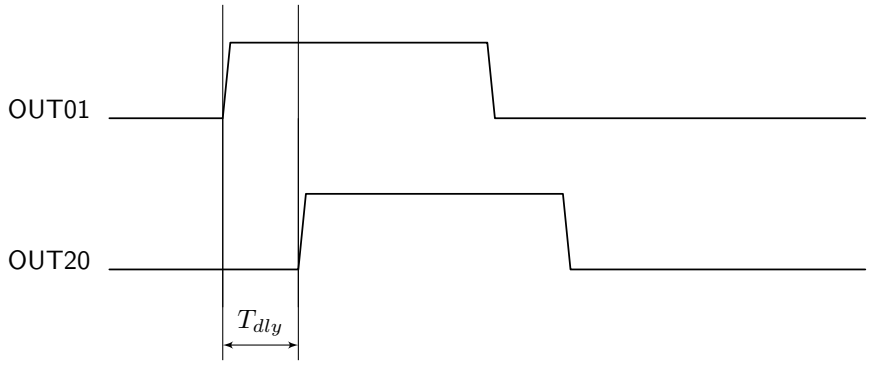
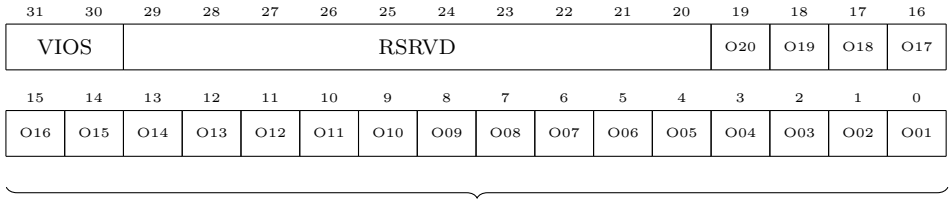


Table 5: Example square waveform parameters (duty = 50%)

Frequency [Hz]	PRDx = TFALLx	TRISEx
4000.000	250	125
2000.000	500	250
1333.333	750	375
1000.000	1000	500
800.000	1250	625
666.667	1500	750
571.429	1750	875
500.000	2000	1000
444.444	2250	1125
400.000	2500	1250
363.636	2750	1375
333.333	3000	1500
307.692	3250	1625
285.714	3500	1750
266.667	3750	1875
250.000	4000	2000
235.294	4250	2125
222.222	4500	2250
210.526	4750	2375
200.000	5000	2500
190.476	5250	2625
181.818	5500	2750
173.913	5750	2875
166.667	6000	3000
160.000	6250	3125
153.846	6500	3250
148.148	6750	3375
142.857	7000	3500
137.931	7250	3625
133.333	7500	3750
129.032	7750	3875
125.000	8000	4000
121.212	8250	4125
117.647	8500	4250
114.286	8750	4375
111.111	9000	4500
108.108	9250	4625
105.263	9500	4750
102.564	9750	4875
100.000	10000	5000

2.3 Registers

2.3.1 ODEF (#0x200) - Outputs default value register



ODEF (#0x200, RWN, init.: 0x0, min.: 0x0, max.: 0x800FFFFF)

[31:30] VIOS — This field is implemented only in MK1.2+ devices. In earlier devices this field is reserved and should not be used (it should be set to zero on writes and ignored on reads).

IO voltage selection:

- 00_b (default) - 3.3 V
- 10_b - 5 V
- 11_b - invalid
- 01_b - invalid

[29:20] RSRVD — Reserved, do not use.

[19] O20 — `OUT20` default value after startup.

[18] O19 — `OUT19` default value after startup.

[17] O18 — `OUT18` default value after startup.

[16] O17 — `OUT17` default value after startup.

[15] O16 — `OUT16` default value after startup.

[14] O15 — `OUT15` default value after startup.

[13] O14 — `OUT14` default value after startup.

[12] O13 — `OUT13` default value after startup.

[11] O12 — `OUT12` default value after startup.

[10] O11 — `OUT11` default value after startup.

[9] O10 — `OUT10` default value after startup.

[8] O09 — `OUT09` default value after startup.

[7] O08 — `OUT08` default value after startup.

[6] O07 — `OUT07` default value after startup.

[5] O06 — `OUT06` default value after startup.

[4] O05 — `OUT05` default value after startup.

[3] O04 — `OUT04` default value after startup.

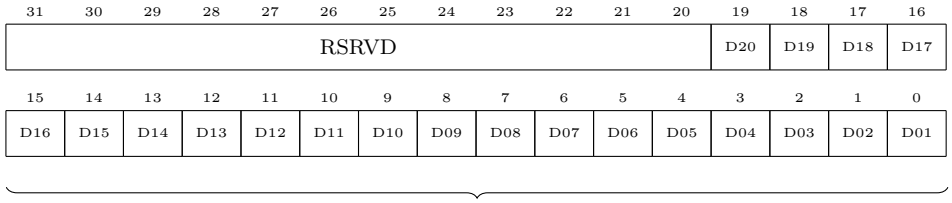
[2] O03 — `OUT03` default value after startup.

[1] O02 — `OUT02` default value after startup.

[0] O01 — `OUT01` default value after startup.

2.3.2 ODIS (#0x201) - Outputs disable register

! → For MK1, all outputs are disabled at once, therefore the only valid value for this register is either 0x0 or 0xFFFFF. This behavior can change for future hardware revisions.



ODIS (#0x201, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFF)

[31:20] RSRVD — Reserved, do not use.

[19] D20 — `OUT20` disable bit (output enters high impedance state).

[18] D19 — `OUT19` disable bit (output enters high impedance state).

[17] D18 — `OUT18` disable bit (output enters high impedance state).

[16] D17 — `OUT17` disable bit (output enters high impedance state).

[15] D16 — `OUT16` disable bit (output enters high impedance state).

[14] D15 — `OUT15` disable bit (output enters high impedance state).

[13] D14 — `OUT14` disable bit (output enters high impedance state).

[12] D13 — `OUT13` disable bit (output enters high impedance state).

[11] D12 — `OUT12` disable bit (output enters high impedance state).

[10] D11 — `OUT11` disable bit (output enters high impedance state).

[9] D10 — `OUT10` disable bit (output enters high impedance state).

[8] D09 — `OUT09` disable bit (output enters high impedance state).

[7] D08 — `OUT08` disable bit (output enters high impedance state).

[6] D07 — `OUT07` disable bit (output enters high impedance state).

[5] D06 — `OUT06` disable bit (output enters high impedance state).

[4] D05 — `OUT05` disable bit (output enters high impedance state).

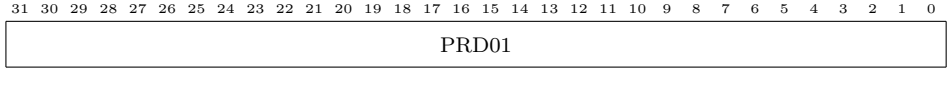
[3] D04 — `OUT04` disable bit (output enters high impedance state).

[2] D03 — `OUT03` disable bit (output enters high impedance state).

[1] D02 — `OUT02` disable bit (output enters high impedance state).

[0] D01 — `OUT01` disable bit (output enters high impedance state).

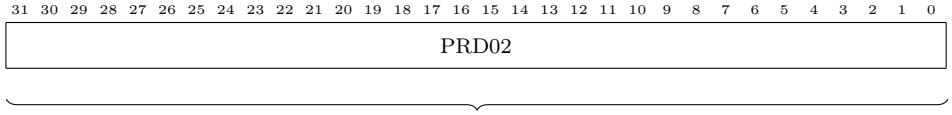
2.3.3 PRD01 (#0x202) - Channel 01 period register



PRD01 (#0x202, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD01 — Channel 01 waveform period (microseconds); set to 0 to disable the channel.

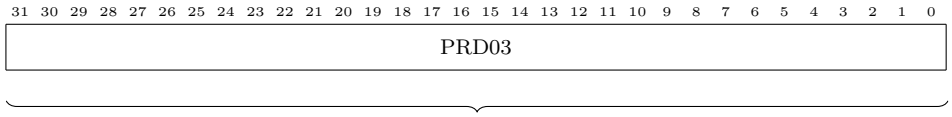
2.3.4 PRD02 (#0x203) - Channel 02 period register



PRD02 (#0x203, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD02 Channel 02 waveform period (microseconds); set to 0 to disable the channel.

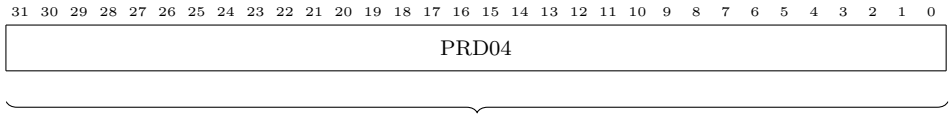
2.3.5 PRD03 (#0x204) - Channel 03 period register



PRD03 (#0x204, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD03 Channel 03 waveform period (microseconds); set to 0 to disable the channel.

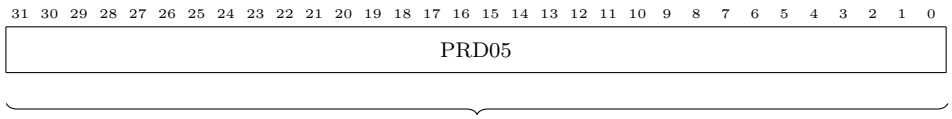
2.3.6 PRD04 (#0x205) - Channel 04 period register



PRD04 (#0x205, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD04 Channel 04 waveform period (microseconds); set to 0 to disable the channel.

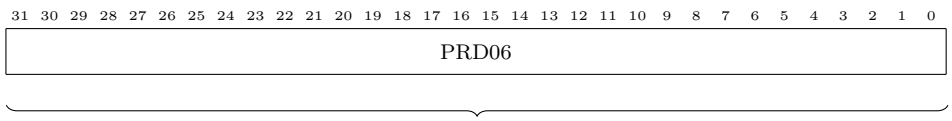
2.3.7 PRD05 (#0x206) - Channel 05 period register



PRD05 (#0x206, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD05 Channel 05 waveform period (microseconds); set to 0 to disable the channel.

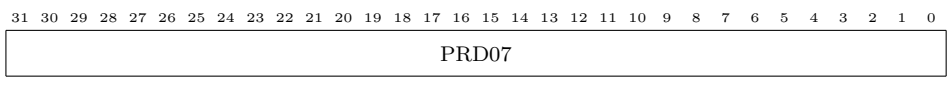
2.3.8 PRD06 (#0x207) - Channel 06 period register



PRD06 (#0x207, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD06 Channel 06 waveform period (microseconds); set to 0 to disable the channel.

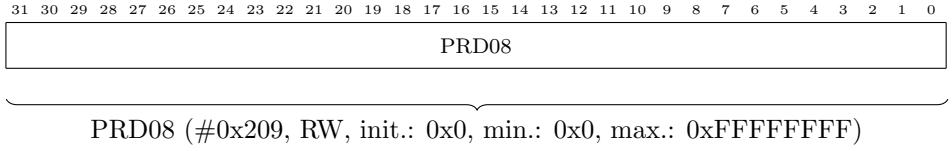
2.3.9 PRD07 (#0x208) - Channel 07 period register



PRD07 (#0x208, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

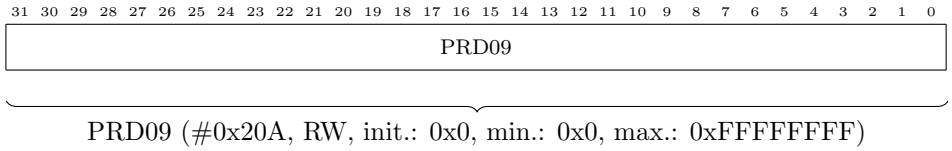
[31:0] PRD07 Channel 07 waveform period (microseconds); set to 0 to disable the channel.

2.3.10 PRD08 (#0x209) - Channel 08 period register



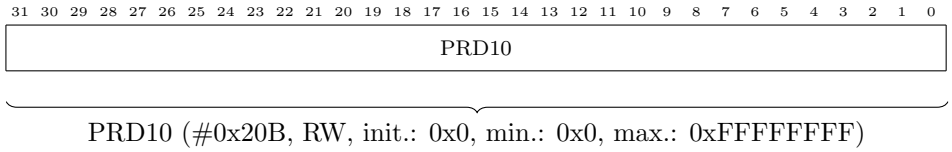
[31:0] PRD08 Channel 08 waveform period (microseconds); set to 0 to disable the channel.

2.3.11 PRD09 (#0x20A) - Channel 09 period register



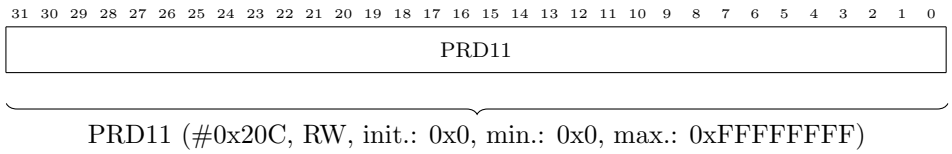
[31:0] PRD09 Channel 09 waveform period (microseconds); set to 0 to disable the channel.

2.3.12 PRD10 (#0x20B) - Channel 10 period register



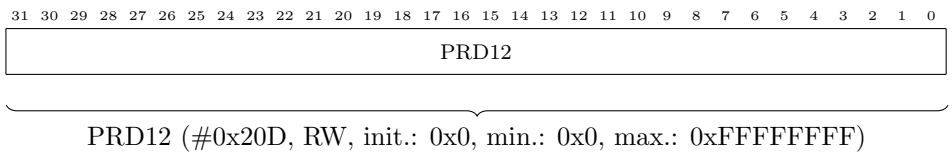
[31:0] PRD10 Channel 10 waveform period (microseconds); set to 0 to disable the channel.

2.3.13 PRD11 (#0x20C) - Channel 11 period register



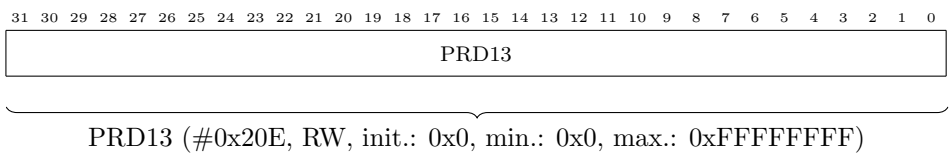
[31:0] PRD11 Channel 11 waveform period (microseconds); set to 0 to disable the channel.

2.3.14 PRD12 (#0x20D) - Channel 12 period register



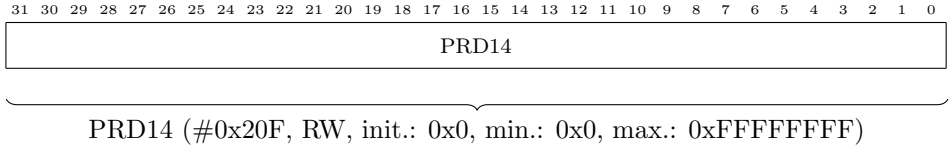
[31:0] PRD12 Channel 12 waveform period (microseconds); set to 0 to disable the channel.

2.3.15 PRD13 (#0x20E) - Channel 13 period register



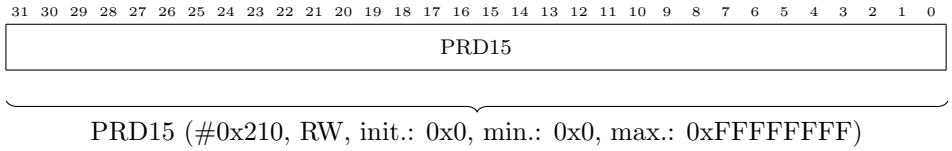
[31:0] PRD13 Channel 13 waveform period (microseconds); set to 0 to disable the channel.

2.3.16 PRD14 (#0x20F) - Channel 14 period register



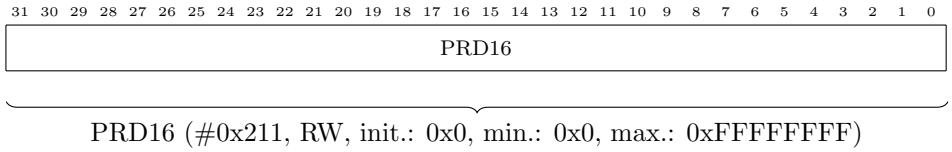
[31:0] PRD14 — Channel 14 waveform period (microseconds); set to 0 to disable the channel.

2.3.17 PRD15 (#0x210) - Channel 15 period register



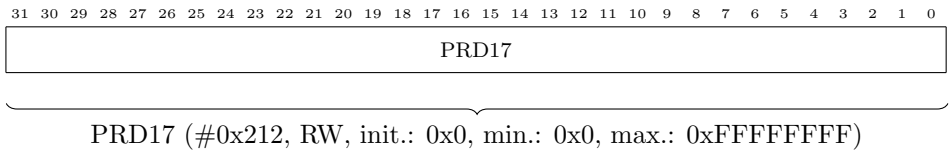
[31:0] PRD15 — Channel 15 waveform period (microseconds); set to 0 to disable the channel.

2.3.18 PRD16 (#0x211) - Channel 16 period register



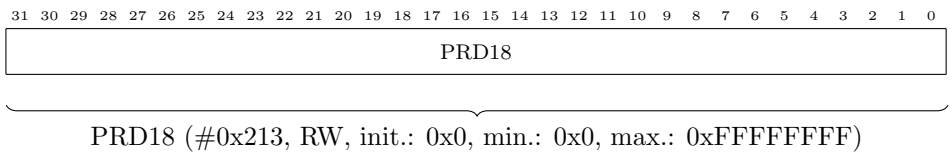
[31:0] PRD16 — Channel 16 waveform period (microseconds); set to 0 to disable the channel.

2.3.19 PRD17 (#0x212) - Channel 17 period register



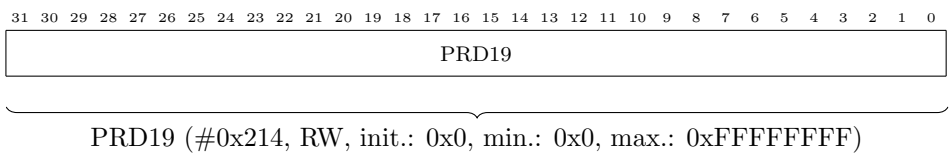
[31:0] PRD17 — Channel 17 waveform period (microseconds); set to 0 to disable the channel.

2.3.20 PRD18 (#0x213) - Channel 18 period register



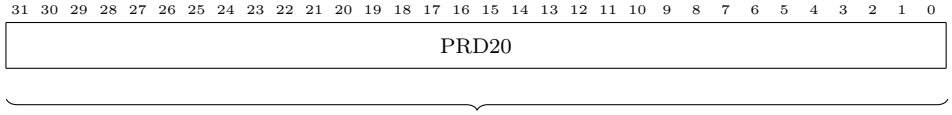
[31:0] PRD18 — Channel 18 waveform period (microseconds); set to 0 to disable the channel.

2.3.21 PRD19 (#0x214) - Channel 19 period register



[31:0] PRD19 — Channel 19 waveform period (microseconds); set to 0 to disable the channel.

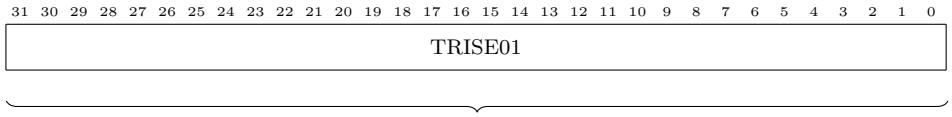
2.3.22 PRD20 (#0x215) - Channel 20 period register



PRD20 (#0x215, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD20 Channel 20 waveform period (microseconds); set to 0 to disable the channel.

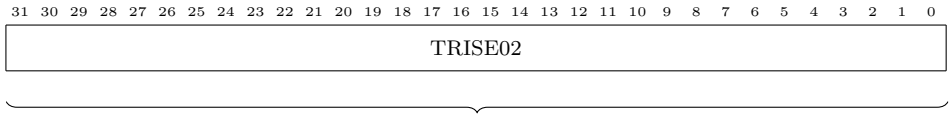
2.3.23 TRISE01 (#0x216) - Channel 01 rise time register



TRISE01 (#0x216, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE01 Channel 01 signal rise time (microseconds).

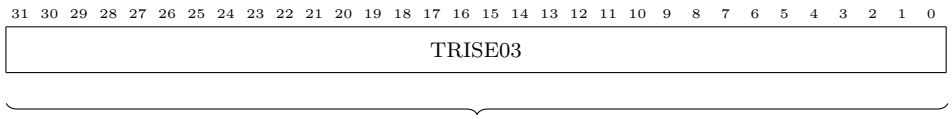
2.3.24 TRISE02 (#0x217) - Channel 02 rise time register



TRISE02 (#0x217, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE02 Channel 02 signal rise time (microseconds).

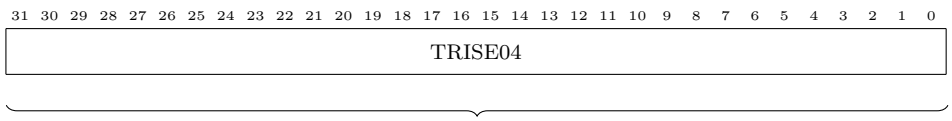
2.3.25 TRISE03 (#0x218) - Channel 03 rise time register



TRISE03 (#0x218, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE03 Channel 03 signal rise time (microseconds).

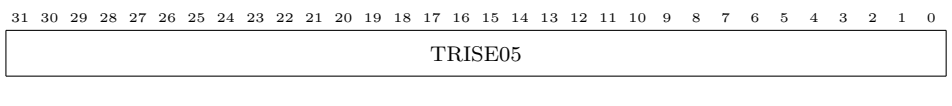
2.3.26 TRISE04 (#0x219) - Channel 04 rise time register



TRISE04 (#0x219, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE04 Channel 04 signal rise time (microseconds).

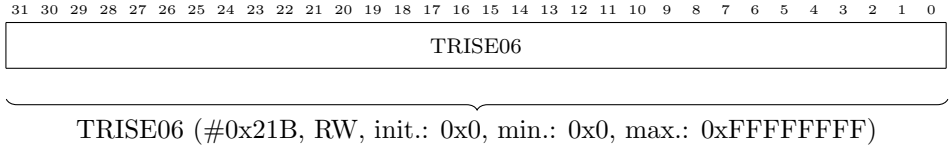
2.3.27 TRISE05 (#0x21A) - Channel 05 rise time register



TRISE05 (#0x21A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

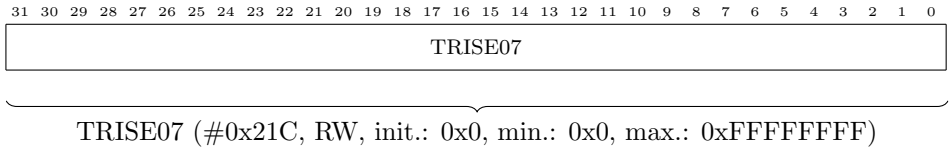
[31:0] TRISE05 Channel 05 signal rise time (microseconds).

2.3.28 TRISE06 (#0x21B) - Channel 06 rise time register



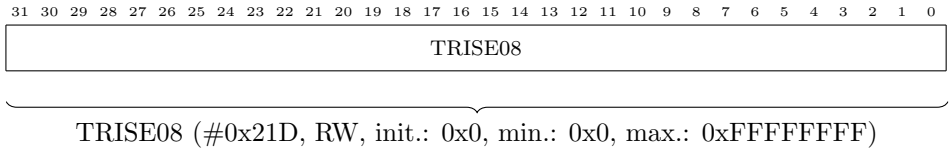
[31:0] TRISE06 Channel 06 signal rise time (microseconds).

2.3.29 TRISE07 (#0x21C) - Channel 07 rise time register



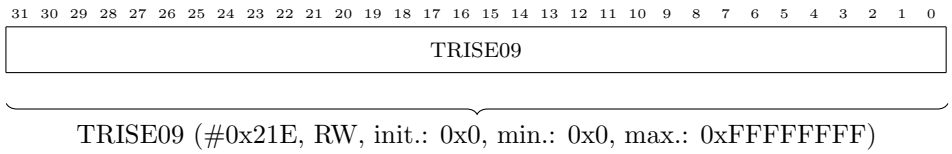
[31:0] TRISE07 Channel 07 signal rise time (microseconds).

2.3.30 TRISE08 (#0x21D) - Channel 08 rise time register



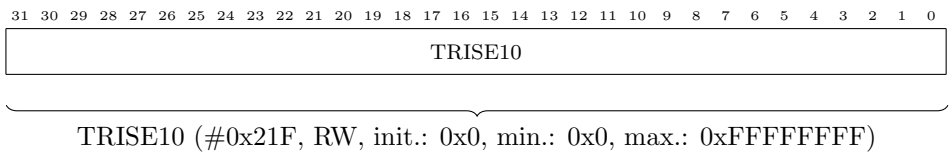
[31:0] TRISE08 Channel 08 signal rise time (microseconds).

2.3.31 TRISE09 (#0x21E) - Channel 09 rise time register



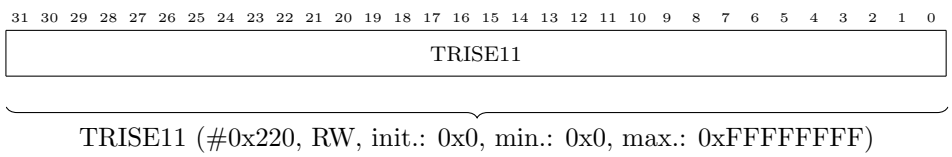
[31:0] TRISE09 Channel 09 signal rise time (microseconds).

2.3.32 TRISE10 (#0x21F) - Channel 10 rise time register



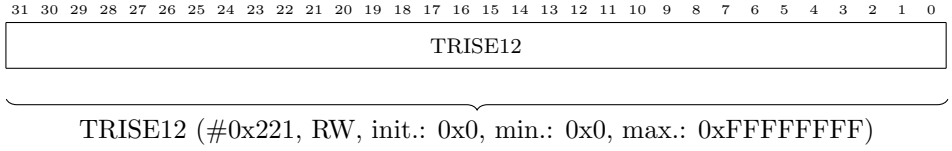
[31:0] TRISE10 Channel 10 signal rise time (microseconds).

2.3.33 TRISE11 (#0x220) - Channel 11 rise time register



[31:0] TRISE11 Channel 11 signal rise time (microseconds).

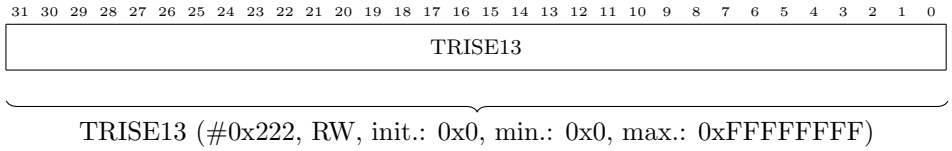
2.3.34 TRISE12 (#0x221) - Channel 12 rise time register



TRISE12 (#0x221, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE12 Channel 12 signal rise time (microseconds).

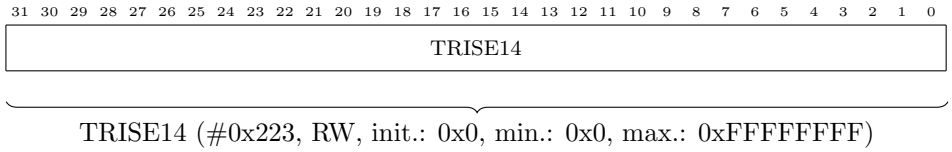
2.3.35 TRISE13 (#0x222) - Channel 13 rise time register



TRISE13 (#0x222, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE13 Channel 13 signal rise time (microseconds).

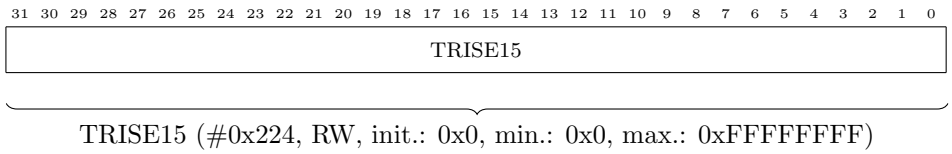
2.3.36 TRISE14 (#0x223) - Channel 14 rise time register



TRISE14 (#0x223, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE14 Channel 14 signal rise time (microseconds).

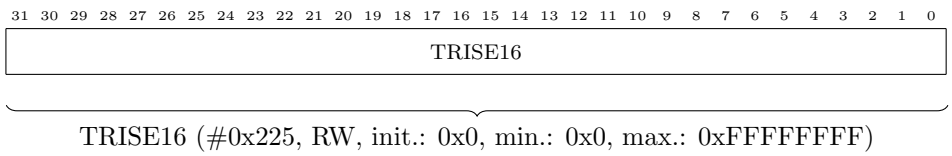
2.3.37 TRISE15 (#0x224) - Channel 15 rise time register



TRISE15 (#0x224, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE15 Channel 15 signal rise time (microseconds).

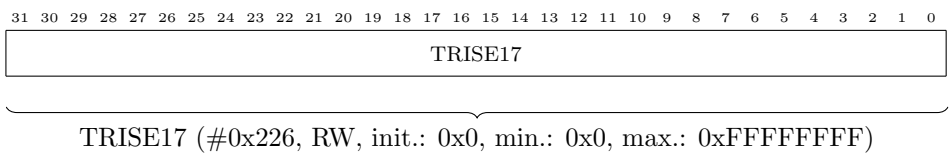
2.3.38 TRISE16 (#0x225) - Channel 16 rise time register



TRISE16 (#0x225, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE16 Channel 16 signal rise time (microseconds).

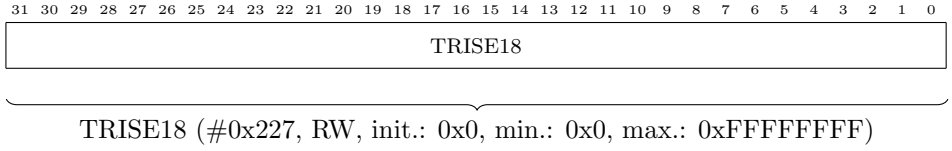
2.3.39 TRISE17 (#0x226) - Channel 17 rise time register



TRISE17 (#0x226, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

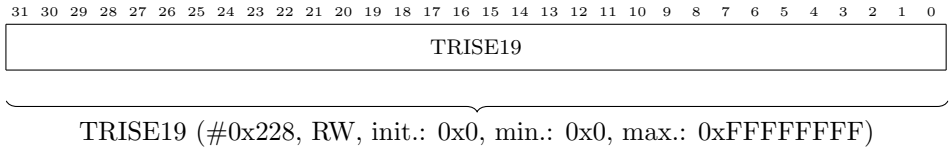
[31:0] TRISE17 Channel 17 signal rise time (microseconds).

2.3.40 TRISE18 (#0x227) - Channel 18 rise time register



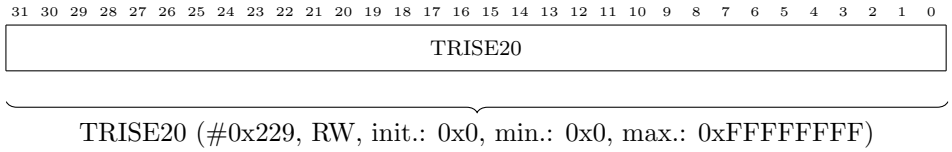
[31:0] TRISE18 Channel 18 signal rise time (microseconds).

2.3.41 TRISE19 (#0x228) - Channel 19 rise time register



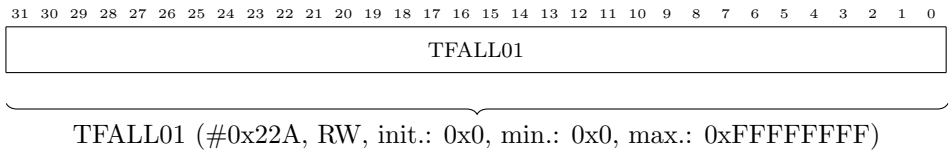
[31:0] TRISE19 Channel 19 signal rise time (microseconds).

2.3.42 TRISE20 (#0x229) - Channel 20 rise time register



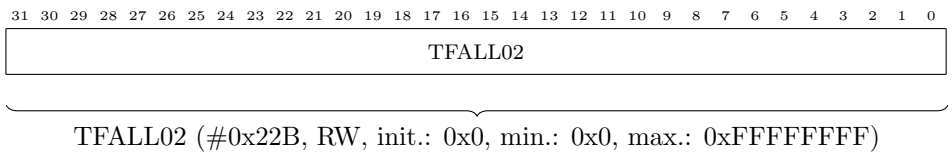
[31:0] TRISE20 Channel 20 signal rise time (microseconds).

2.3.43 TFALL01 (#0x22A) - Channel 01 fall time register



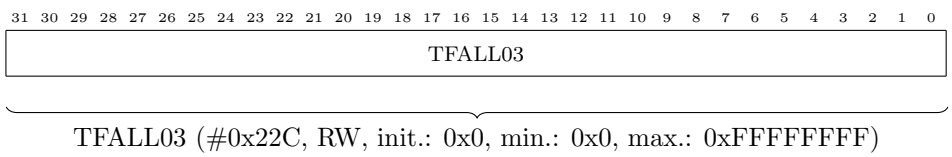
[31:0] TFALL01 Channel 01 signal fall time (microseconds).

2.3.44 TFALL02 (#0x22B) - Channel 02 fall time register



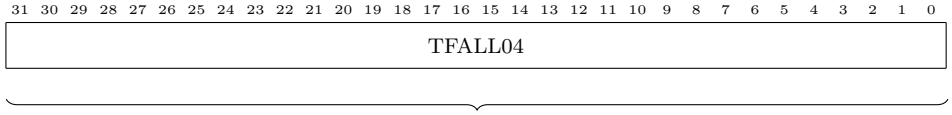
[31:0] TFALL02 Channel 02 signal fall time (microseconds).

2.3.45 TFALL03 (#0x22C) - Channel 03 fall time register



[31:0] TFALL03 Channel 03 signal fall time (microseconds).

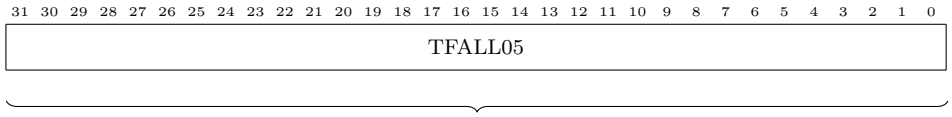
2.3.46 TFALL04 (#0x22D) - Channel 04 fall time register



TFALL04 (#0x22D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL04 Channel 04 signal fall time (microseconds).

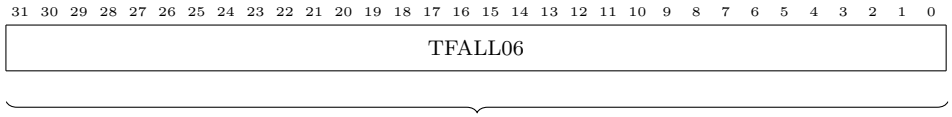
2.3.47 TFALL05 (#0x22E) - Channel 05 fall time register



TFALL05 (#0x22E, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL05 Channel 05 signal fall time (microseconds).

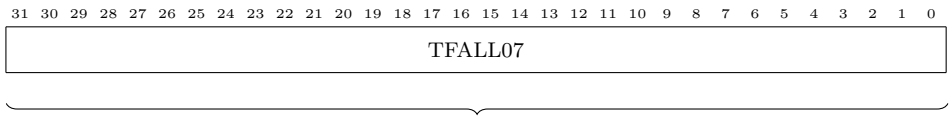
2.3.48 TFALL06 (#0x22F) - Channel 06 fall time register



TFALL06 (#0x22F, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL06 Channel 06 signal fall time (microseconds).

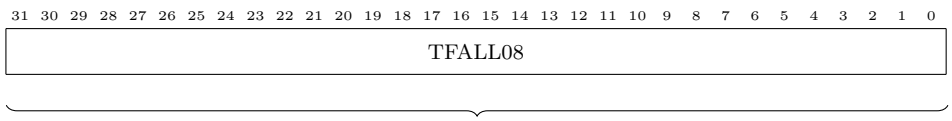
2.3.49 TFALL07 (#0x230) - Channel 07 fall time register



TFALL07 (#0x230, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL07 Channel 07 signal fall time (microseconds).

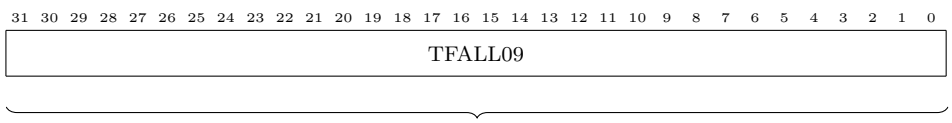
2.3.50 TFALL08 (#0x231) - Channel 08 fall time register



TFALL08 (#0x231, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL08 Channel 08 signal fall time (microseconds).

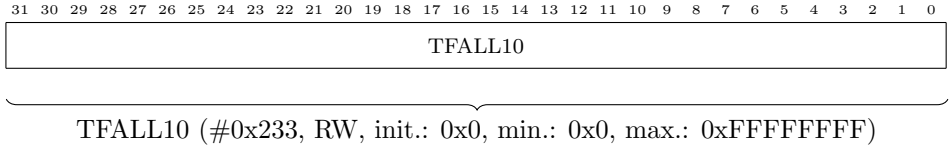
2.3.51 TFALL09 (#0x232) - Channel 09 fall time register



TFALL09 (#0x232, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL09 Channel 09 signal fall time (microseconds).

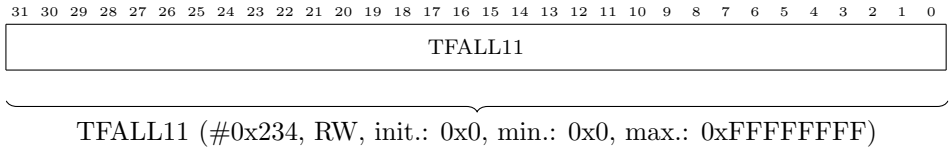
2.3.52 TFALL10 (#0x233) - Channel 10 fall time register



TFALL10 (#0x233, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL10 Channel 10 signal fall time (microseconds).

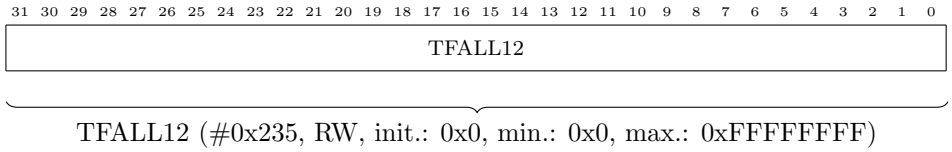
2.3.53 TFALL11 (#0x234) - Channel 11 fall time register



TFALL11 (#0x234, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL11 Channel 11 signal fall time (microseconds).

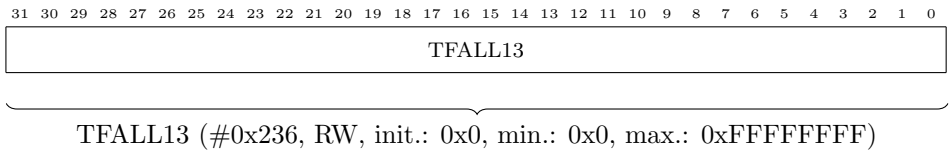
2.3.54 TFALL12 (#0x235) - Channel 12 fall time register



TFALL12 (#0x235, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL12 Channel 12 signal fall time (microseconds).

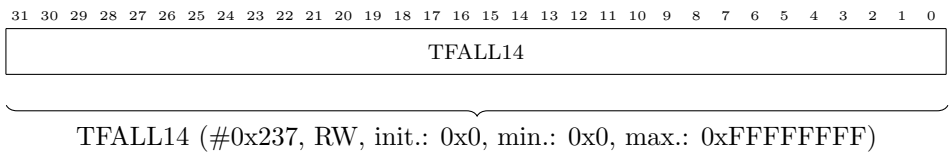
2.3.55 TFALL13 (#0x236) - Channel 13 fall time register



TFALL13 (#0x236, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL13 Channel 13 signal fall time (microseconds).

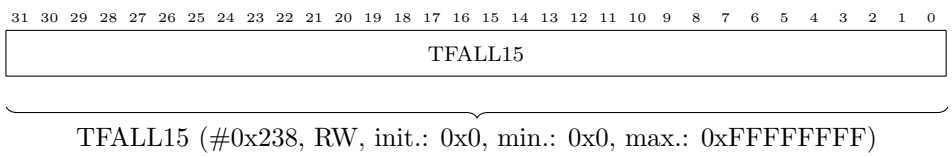
2.3.56 TFALL14 (#0x237) - Channel 14 fall time register



TFALL14 (#0x237, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL14 Channel 14 signal fall time (microseconds).

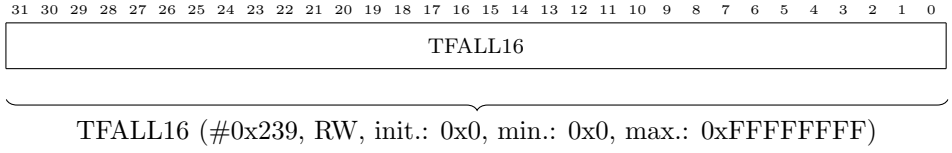
2.3.57 TFALL15 (#0x238) - Channel 15 fall time register



TFALL15 (#0x238, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL15 Channel 15 signal fall time (microseconds).

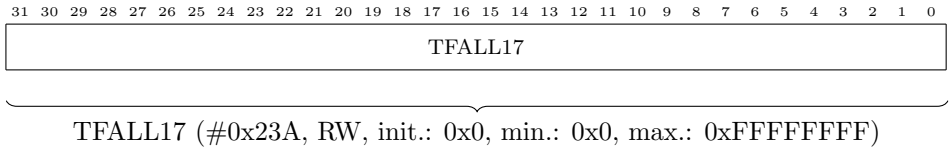
2.3.58 TFALL16 (#0x239) - Channel 16 fall time register



TFALL16 (#0x239, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL16 Channel 16 signal fall time (microseconds).

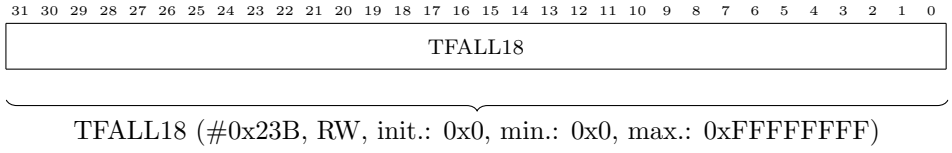
2.3.59 TFALL17 (#0x23A) - Channel 17 fall time register



TFALL17 (#0x23A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL17 Channel 17 signal fall time (microseconds).

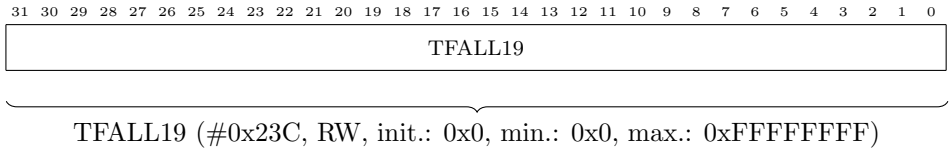
2.3.60 TFALL18 (#0x23B) - Channel 18 fall time register



TFALL18 (#0x23B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL18 Channel 18 signal fall time (microseconds).

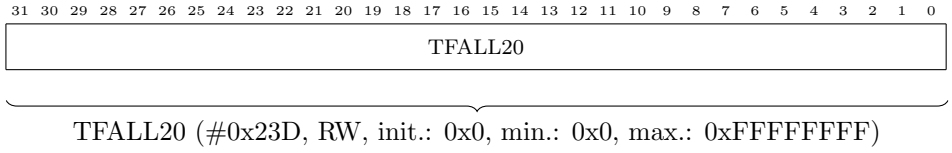
2.3.61 TFALL19 (#0x23C) - Channel 19 fall time register



TFALL19 (#0x23C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL19 Channel 19 signal fall time (microseconds).

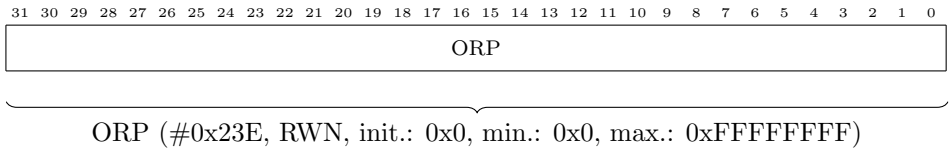
2.3.62 TFALL20 (#0x23D) - Channel 20 fall time register



TFALL20 (#0x23D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL20 Channel 20 signal fall time (microseconds).

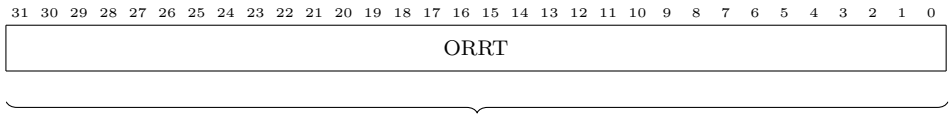
2.3.63 ORP (#0x23E) - Outputs restore period



ORP (#0x23E, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] ORP Outputs restore period (microseconds). When set to a non-zero value, defines the time after which all outputs are set to their default values if no output setting command is executed. Writing this register causes ORRT register to be reloaded.

2.3.64 ORRT (#0x23F) - Outptus restore remaining time register



ORRT (#0x23F, RO, init.: 0x0)

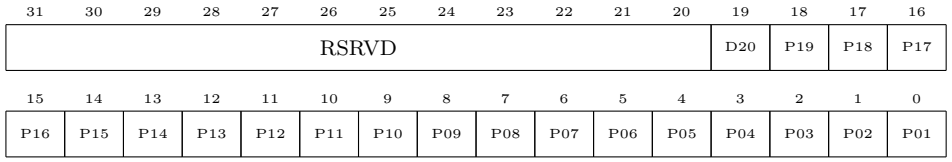
[31:0] ORRT — Outputs restore remaining time (microseconds). Represents the time remaining before all outputs are restored to their default value. This register is reloaded with the value of ORP each time the outputs state is being set (and also when ORP is written). When ORP is non-zero and the value of ORRT drops to zero, the following actions are performed:

- all signal generators are disabled by setting their PRDx registers to 0;
- the outputs are set to the value of ODEF register.

2.3.65 PSEL (#0x240) - Pull-up / pull-down selection register

! → This register is implemented only in MK1.2+ devices.

! → For MK1.2, all pull resistors are configured at once, therefore the only valid value for this register is either 0x0 or 0xFFFFF. This behavior can change in future hardware revisions.



PSEL (#0x240, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFF)

[31:20] RSRVD — Reserved, do not use.

[19] D20 — `IN20` pull-up (1) or pull-down (0) selection bit.

[18] P19 — `IN19` pull-up (1) or pull-down (0) selection bit.

[17] P18 — `IN18` pull-up (1) or pull-down (0) selection bit.

[16] P17 — `IN17` pull-up (1) or pull-down (0) selection bit.

[15] P16 — `IN16` pull-up (1) or pull-down (0) selection bit.

[14] P15 — `IN15` pull-up (1) or pull-down (0) selection bit.

[13] P14 — `IN14` pull-up (1) or pull-down (0) selection bit.

[12] P13 — `IN13` pull-up (1) or pull-down (0) selection bit.

[11] P12 — `IN12` pull-up (1) or pull-down (0) selection bit.

[10] P11 — `IN11` pull-up (1) or pull-down (0) selection bit.

[9] P10 — `IN10` pull-up (1) or pull-down (0) selection bit.

[8] P09 — `IN09` pull-up (1) or pull-down (0) selection bit.

[7] P08 — `IN08` pull-up (1) or pull-down (0) selection bit.

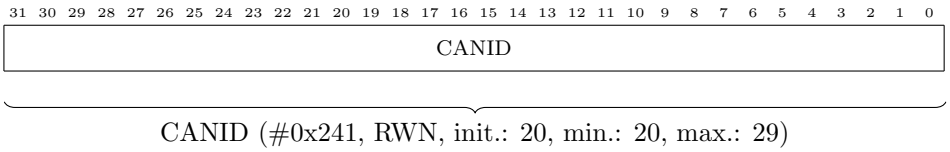
[6] P07 — `IN07` pull-up (1) or pull-down (0) selection bit.

[5] P06 — `IN06` pull-up (1) or pull-down (0) selection bit.

[4]	P05	IN05 pull-up (1) or pull-down (0) selection bit.
[3]	P04	IN04 pull-up (1) or pull-down (0) selection bit.
[2]	P03	IN03 pull-up (1) or pull-down (0) selection bit.
[1]	P02	IN02 pull-up (1) or pull-down (0) selection bit.
[0]	P01	IN01 pull-up (1) or pull-down (0) selection bit.

2.3.66 CANID (#0x241) - The node's CAN ID / address register

Writing to this register will change the node's address. The CANopen node will be stopped, initialized and then will automatically enter operational mode using the new address.



CANID (#0x241, RWN, init.: 20, min.: 20, max.: 29)

[31:0]	CANID	The CAN node ID. See Setting the CAN node address for details.

IV. MATES-DAC5-MK1 - analogue outputs module

This module is used to generate analogue signals. It provides 40 DAC channels with the voltage range of 0 to 5 V.

1 Properties

1.1 Electrical parameters

Table 6: Device parameters

Parameter	Symbol	Min.	Typ.	Max.
Current consumption[a]	I_N	-	-	100 mA

[a] at V_{IN} = 24 V, no load

Table 7: Analogue outputs OUT01 - OUT40

Parameter	Symbol	Min.	Typ.	Max.
Output voltage	V_O	0 V	-	5 V
Output current	I_O	-	-	±20 mA
Resolution	N	-	2^{16}	-
Crosstalk	V_C	-	-	±20 μV
DC accuracy[a]	Δ_V	-	-	±10 mV
Noise[b]	V_N	-	-	100 mV p-p

[a] guaranteed within 6 months after calibration
[b] in 1 MHz bandwidth

STOP → The DAC outputs are not short-circuit resistant.

2 Programming

2.1 Datagrams

2.1.1 Standard status - HEARTBEAT

7	6	5	4	3	2	1	0
STATUS							
HEARTBEAT (NMT = 700_h+ node number)							

[7:0] STATUS — Communication status of node, according to CANopen.

2.1.2 Analogue output value - DAC5-OUT

This datagram is used to set the output value for an individual DAC output. The datagram can be read back using remote frame. This datagram directly writes the DAC registers bypassing the calibration data. The theoretical voltage output value can be calculated from the binary value as follows:

$$V_{DAC} = 5\frac{X}{2^{16}} \tag{4}$$

Where:

V_{DAC}: the DAC output voltage in Volts

X: the binary value sent to the DAC

2^{16}: the DAC resolution

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RO	CHANNEL							LSB								MSB								RSVD							
byte 0								byte 1								byte 2								byte 3							
DAC5-OUT (PDO 1 RX = 200_h+ node number)																															

[0] RO — Read only operation. Set this bit to 1 and the channel field to desired channel and then read the datagram using remote frame to get the actual value of the DAC. Use this bit when random DAC read is required. You can immediately read the lastly written DAC value without sending frame with RO=1.

[1:7] CHANNEL — DAC channel selector (0 means OUT01, 39 means OUT40).

[8:15] LSB — DAC binary output value, less significant byte.

[16:23] MSB — DAC binary output value, more significant byte.

[24:31] RSVD — Reserved for future use.

2.1.3 Analogue output voltage - DAC5-VOUT

This datagram is used to set the output voltage for an individual DAC output. The datagram can be read back using remote frame. The provided voltage value is identical to the actual value at the output. The written DAC value takes the calibration data into account.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
RO	CHANNEL							VALUE (LE)																																	
byte 0								byte 1								byte 2								byte 3								byte 4							
DAC5-VOUT (PDO 2 RX = 300_h+ node number)																																							

[0] RO — Read only operation. Set this bit to 1 and the channel field to desired channel and then read the datagram using remote frame to get the actual value of the DAC. Use this bit when random DAC read is required. You can immediately read the lastly written DAC value without sending frame with RO=1.

[1:7] CHANNEL — DAC channel selector (0 means OUT01, 39 means OUT40).

[8:39] VALUE — DAC voltage value [Volts], written as IEEE-754 single precision floating point number in Little Endian.

2.2 Registers

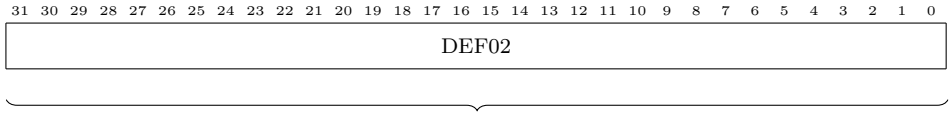
2.2.1 DEF01 (#0x500) - OUT01 default voltage register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEF01																															

DEF01 (#0x500, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF01 — Default output voltage after system start.

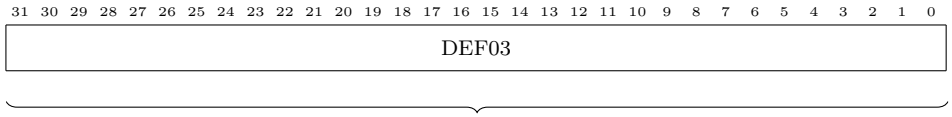
2.2.2 DEF02 (#0x501) - OUT02 default voltage register



DEF02 (#0x501, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF02 Default output voltage after system start.

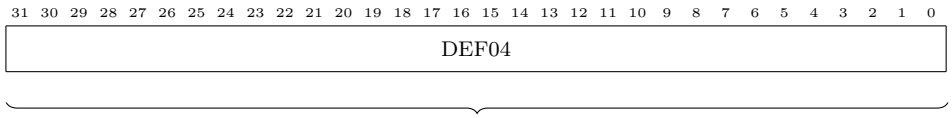
2.2.3 DEF03 (#0x502) - OUT03 default voltage register



DEF03 (#0x502, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF03 Default output voltage after system start.

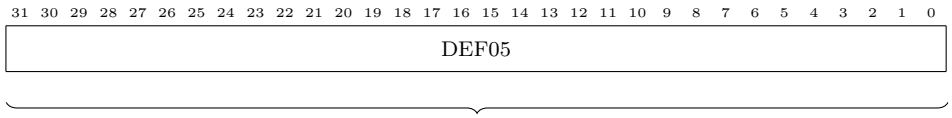
2.2.4 DEF04 (#0x503) - OUT04 default voltage register



DEF04 (#0x503, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF04 Default output voltage after system start.

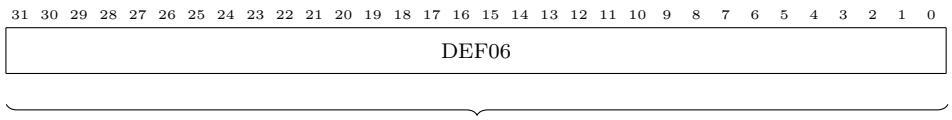
2.2.5 DEF05 (#0x504) - OUT05 default voltage register



DEF05 (#0x504, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF05 Default output voltage after system start.

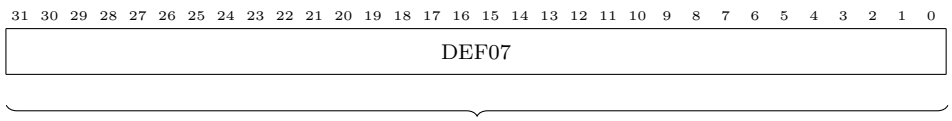
2.2.6 DEF06 (#0x505) - OUT06 default voltage register



DEF06 (#0x505, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF06 Default output voltage after system start.

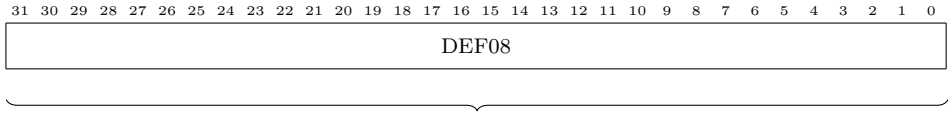
2.2.7 DEF07 (#0x506) - OUT07 default voltage register



DEF07 (#0x506, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF07 Default output voltage after system start.

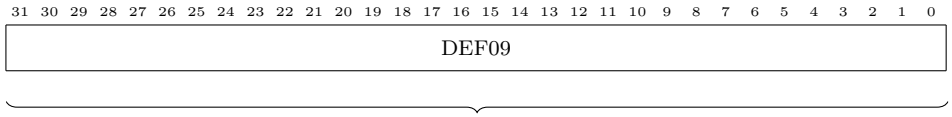
2.2.8 DEF08 (#0x507) - OUT08 default voltage register



DEF08 (#0x507, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF08 Default output voltage after system start.

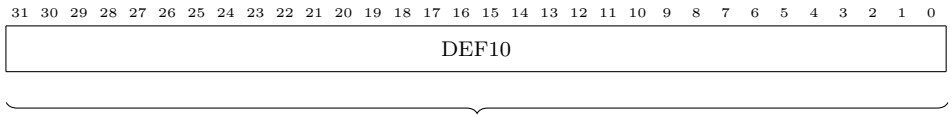
2.2.9 DEF09 (#0x508) - OUT09 default voltage register



DEF09 (#0x508, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF09 Default output voltage after system start.

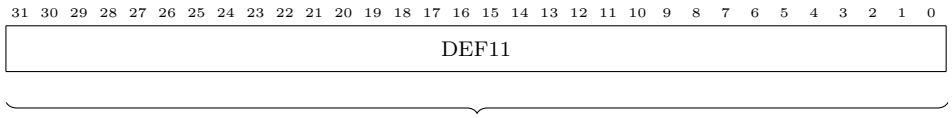
2.2.10 DEF10 (#0x509) - OUT10 default voltage register



DEF10 (#0x509, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF10 Default output voltage after system start.

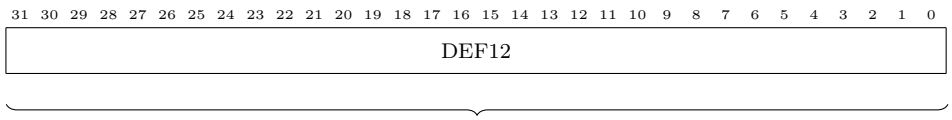
2.2.11 DEF11 (#0x50A) - OUT11 default voltage register



DEF11 (#0x50A, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF11 Default output voltage after system start.

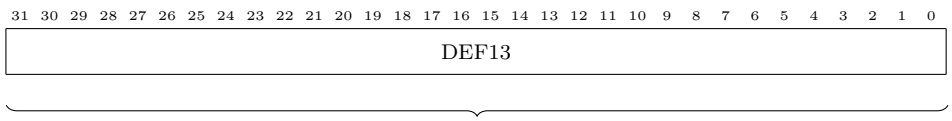
2.2.12 DEF12 (#0x50B) - OUT12 default voltage register



DEF12 (#0x50B, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF12 Default output voltage after system start.

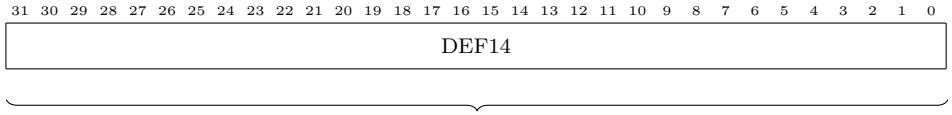
2.2.13 DEF13 (#0x50C) - OUT13 default voltage register



DEF13 (#0x50C, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF13 Default output voltage after system start.

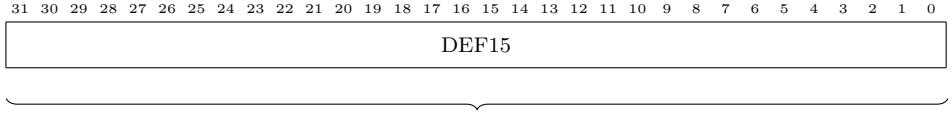
2.2.14 DEF14 (#0x50D) - OUT14 default voltage register



DEF14 (#0x50D, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF14 Default output voltage after system start.

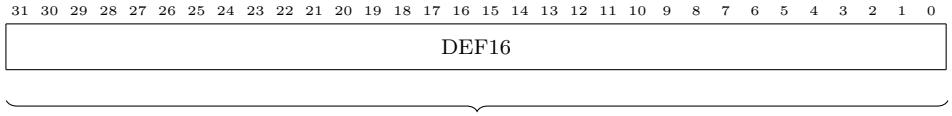
2.2.15 DEF15 (#0x50E) - OUT15 default voltage register



DEF15 (#0x50E, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF15 Default output voltage after system start.

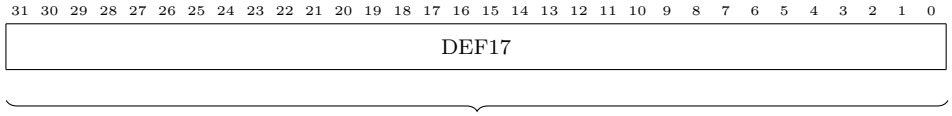
2.2.16 DEF16 (#0x50F) - OUT16 default voltage register



DEF16 (#0x50F, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF16 Default output voltage after system start.

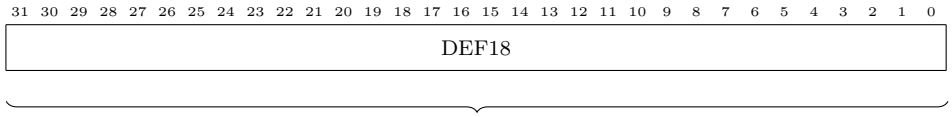
2.2.17 DEF17 (#0x510) - OUT17 default voltage register



DEF17 (#0x510, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF17 Default output voltage after system start.

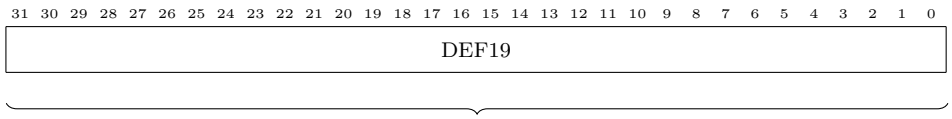
2.2.18 DEF18 (#0x511) - OUT18 default voltage register



DEF18 (#0x511, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF18 Default output voltage after system start.

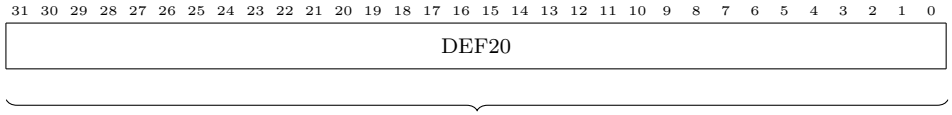
2.2.19 DEF19 (#0x512) - OUT19 default voltage register



DEF19 (#0x512, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF19 Default output voltage after system start.

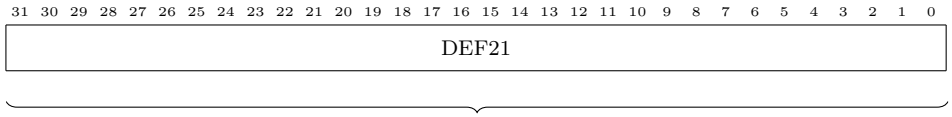
2.2.20 DEF20 (#0x513) - OUT20 default voltage register



DEF20 (#0x513, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF20 Default output voltage after system start.

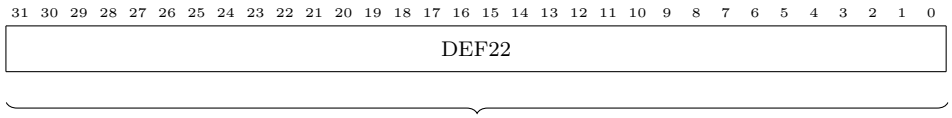
2.2.21 DEF21 (#0x514) - OUT21 default voltage register



DEF21 (#0x514, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF21 Default output voltage after system start.

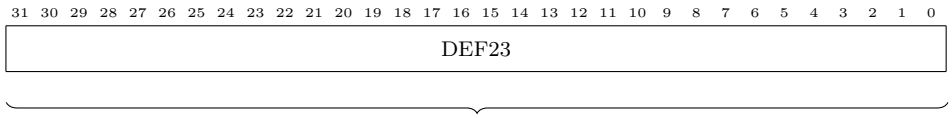
2.2.22 DEF22 (#0x515) - OUT22 default voltage register



DEF22 (#0x515, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF22 Default output voltage after system start.

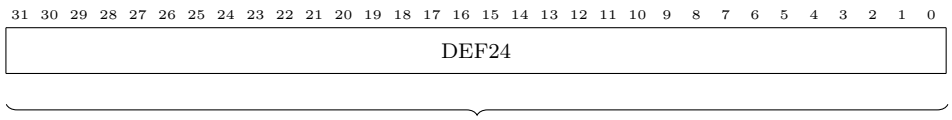
2.2.23 DEF23 (#0x516) - OUT23 default voltage register



DEF23 (#0x516, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF23 Default output voltage after system start.

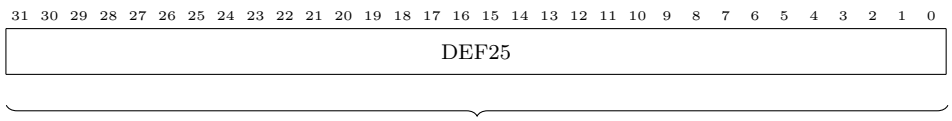
2.2.24 DEF24 (#0x517) - OUT24 default voltage register



DEF24 (#0x517, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF24 Default output voltage after system start.

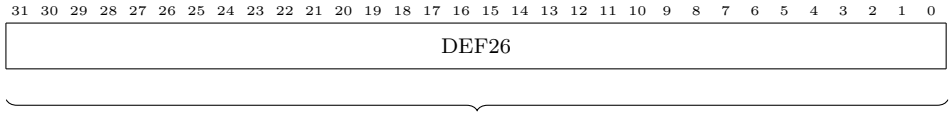
2.2.25 DEF25 (#0x518) - OUT25 default voltage register



DEF25 (#0x518, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF25 Default output voltage after system start.

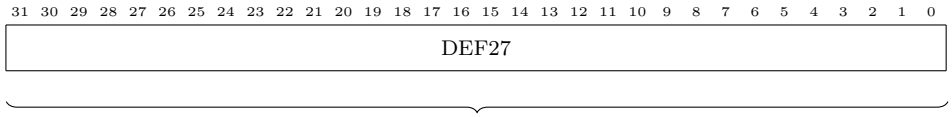
2.2.26 DEF26 (#0x519) - OUT26 default voltage register



DEF26 (#0x519, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF26 Default output voltage after system start.

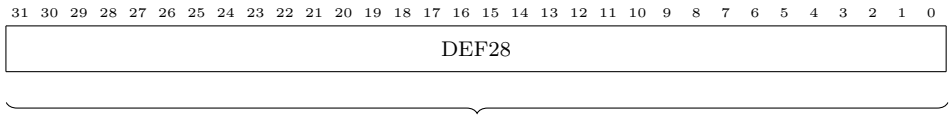
2.2.27 DEF27 (#0x51A) - OUT27 default voltage register



DEF27 (#0x51A, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF27 Default output voltage after system start.

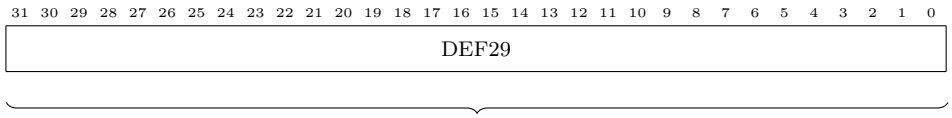
2.2.28 DEF28 (#0x51B) - OUT28 default voltage register



DEF28 (#0x51B, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF28 Default output voltage after system start.

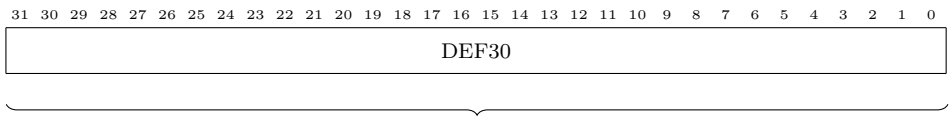
2.2.29 DEF29 (#0x51C) - OUT29 default voltage register



DEF29 (#0x51C, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF29 Default output voltage after system start.

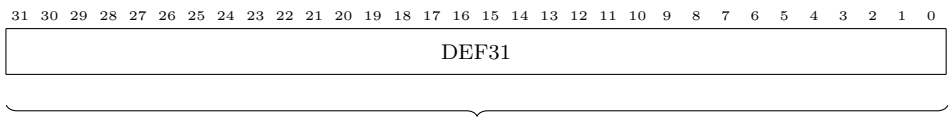
2.2.30 DEF30 (#0x51D) - OUT30 default voltage register



DEF30 (#0x51D, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF30 Default output voltage after system start.

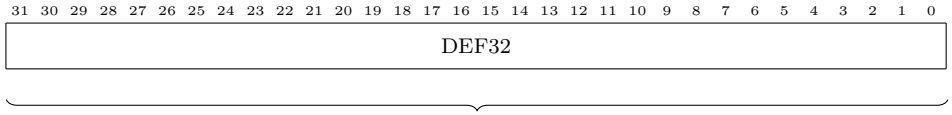
2.2.31 DEF31 (#0x51E) - OUT31 default voltage register



DEF31 (#0x51E, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF31 Default output voltage after system start.

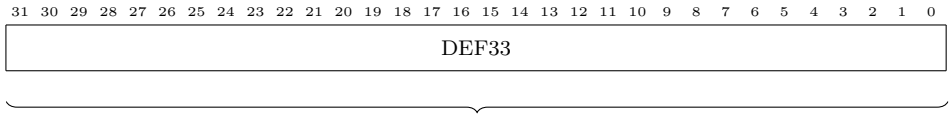
2.2.32 DEF32 (#0x51F) - OUT32 default voltage register



DEF32 (#0x51F, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF32 Default output voltage after system start.

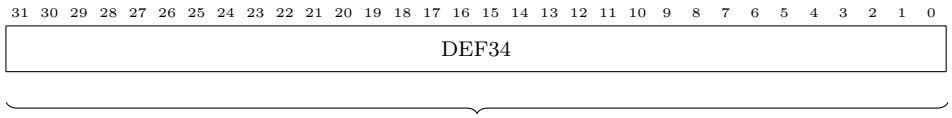
2.2.33 DEF33 (#0x520) - OUT33 default voltage register



DEF33 (#0x520, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF33 Default output voltage after system start.

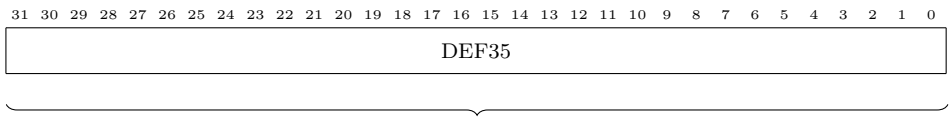
2.2.34 DEF34 (#0x521) - OUT34 default voltage register



DEF34 (#0x521, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF34 Default output voltage after system start.

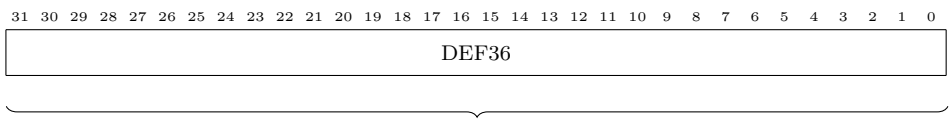
2.2.35 DEF35 (#0x522) - OUT35 default voltage register



DEF35 (#0x522, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF35 Default output voltage after system start.

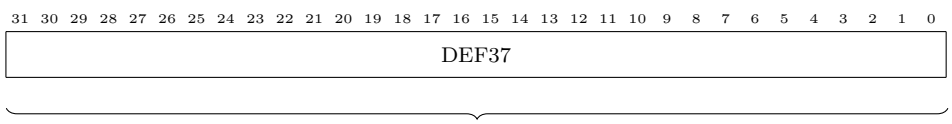
2.2.36 DEF36 (#0x523) - OUT36 default voltage register



DEF36 (#0x523, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF36 Default output voltage after system start.

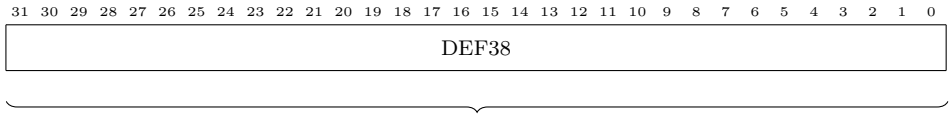
2.2.37 DEF37 (#0x524) - OUT37 default voltage register



DEF37 (#0x524, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF37 Default output voltage after system start.

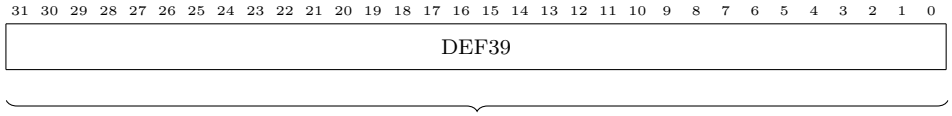
2.2.38 DEF38 (#0x525) - OUT38 default voltage register



DEF38 (#0x525, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF38 Default output voltage after system start.

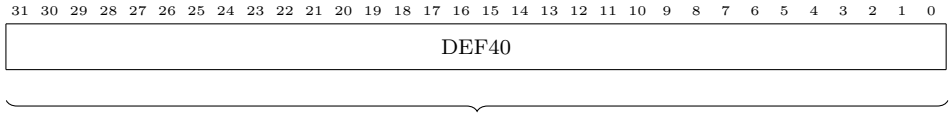
2.2.39 DEF39 (#0x526) - OUT39 default voltage register



DEF39 (#0x526, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF39 Default output voltage after system start.

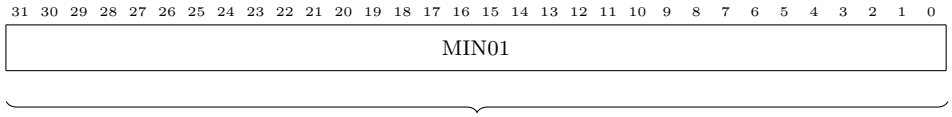
2.2.40 DEF40 (#0x527) - OUT40 default voltage register



DEF40 (#0x527, RWN, init.: 0.1, min.: 0.0, max.: 5.0)

[31:0] DEF40 Default output voltage after system start.

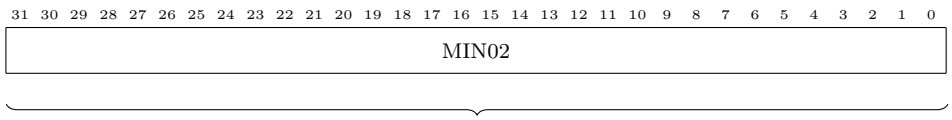
2.2.41 MIN01 (#0x528) - OUT01 minimum voltage register



MIN01 (#0x528, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN01 Minimum voltage (Volts) that can be set for this output.

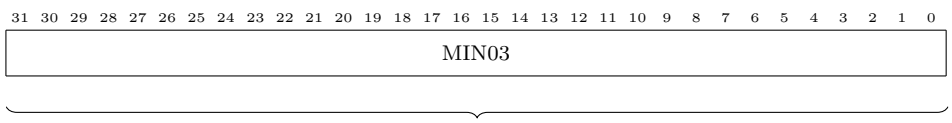
2.2.42 MIN02 (#0x529) - OUT02 minimum voltage register



MIN02 (#0x529, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN02 Minimum voltage (Volts) that can be set for this output.

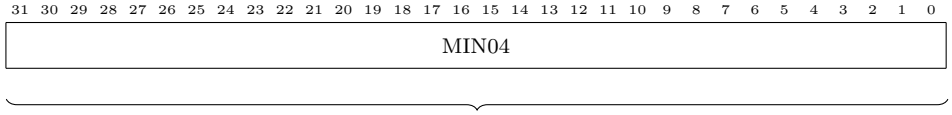
2.2.43 MIN03 (#0x52A) - OUT03 minimum voltage register



MIN03 (#0x52A, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN03 Minimum voltage (Volts) that can be set for this output.

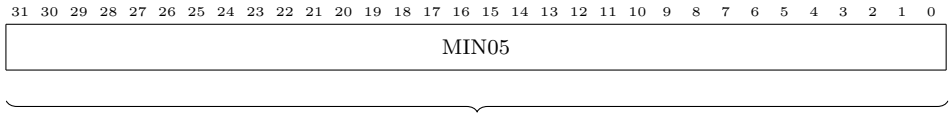
2.2.44 MIN04 (#0x52B) - OUT04 minimum voltage register



MIN04 (#0x52B, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN04 Minimum voltage (Volts) that can be set for this output.

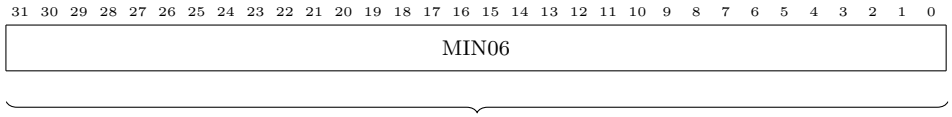
2.2.45 MIN05 (#0x52C) - OUT05 minimum voltage register



MIN05 (#0x52C, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN05 Minimum voltage (Volts) that can be set for this output.

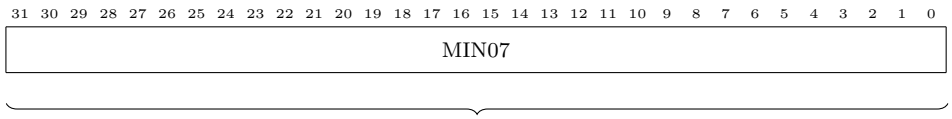
2.2.46 MIN06 (#0x52D) - OUT06 minimum voltage register



MIN06 (#0x52D, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN06 Minimum voltage (Volts) that can be set for this output.

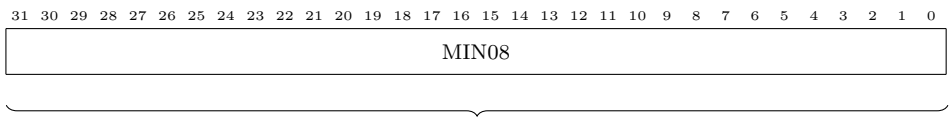
2.2.47 MIN07 (#0x52E) - OUT07 minimum voltage register



MIN07 (#0x52E, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN07 Minimum voltage (Volts) that can be set for this output.

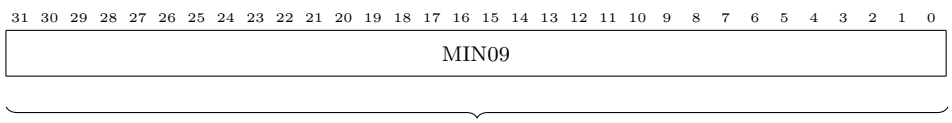
2.2.48 MIN08 (#0x52F) - OUT08 minimum voltage register



MIN08 (#0x52F, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN08 Minimum voltage (Volts) that can be set for this output.

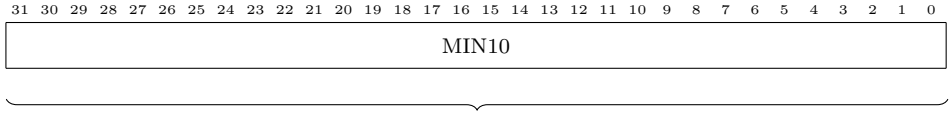
2.2.49 MIN09 (#0x530) - OUT09 minimum voltage register



MIN09 (#0x530, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN09 Minimum voltage (Volts) that can be set for this output.

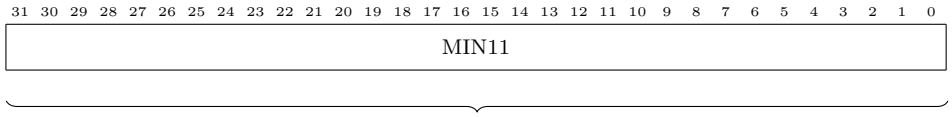
2.2.50 MIN10 (#0x531) - OUT10 minimum voltage register



MIN10 (#0x531, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN10 Minimum voltage (Volts) that can be set for this output.

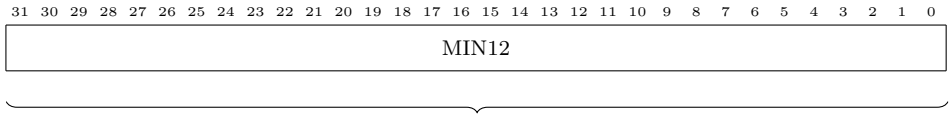
2.2.51 MIN11 (#0x532) - OUT11 minimum voltage register



MIN11 (#0x532, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN11 Minimum voltage (Volts) that can be set for this output.

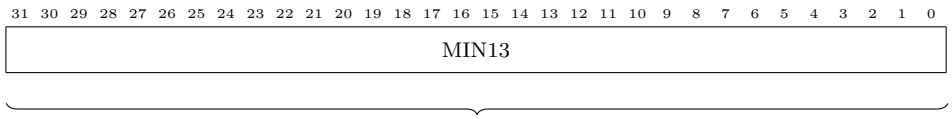
2.2.52 MIN12 (#0x533) - OUT12 minimum voltage register



MIN12 (#0x533, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN12 Minimum voltage (Volts) that can be set for this output.

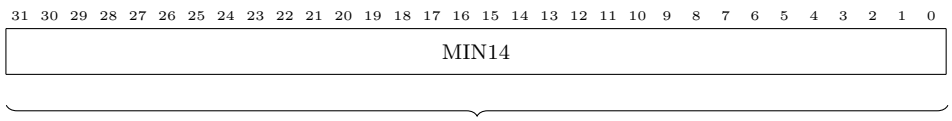
2.2.53 MIN13 (#0x534) - OUT13 minimum voltage register



MIN13 (#0x534, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN13 Minimum voltage (Volts) that can be set for this output.

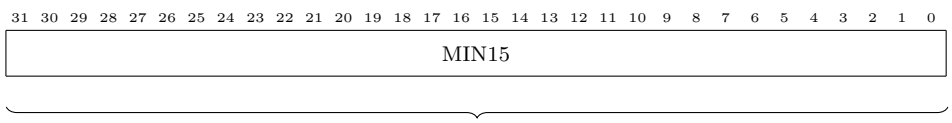
2.2.54 MIN14 (#0x535) - OUT14 minimum voltage register



MIN14 (#0x535, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN14 Minimum voltage (Volts) that can be set for this output.

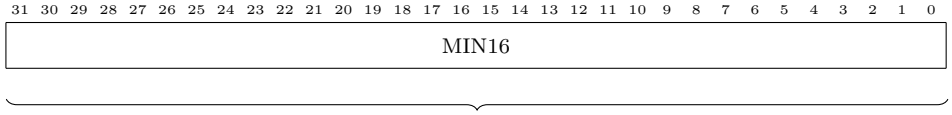
2.2.55 MIN15 (#0x536) - OUT15 minimum voltage register



MIN15 (#0x536, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN15 Minimum voltage (Volts) that can be set for this output.

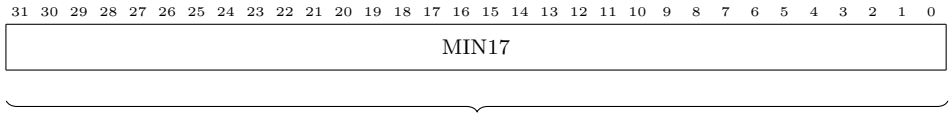
2.2.56 MIN16 (#0x537) - OUT16 minimum voltage register



MIN16 (#0x537, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN16 Minimum voltage (Volts) that can be set for this output.

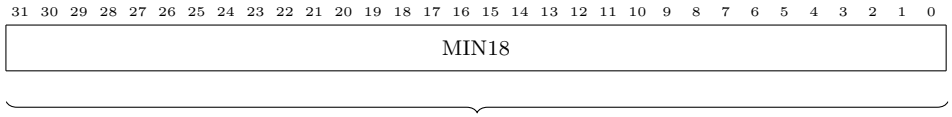
2.2.57 MIN17 (#0x538) - OUT17 minimum voltage register



MIN17 (#0x538, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN17 Minimum voltage (Volts) that can be set for this output.

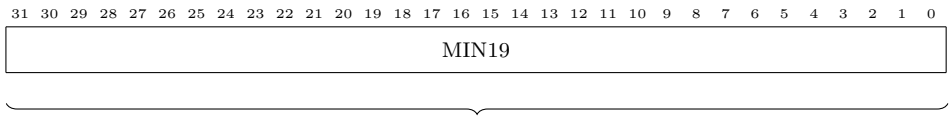
2.2.58 MIN18 (#0x539) - OUT18 minimum voltage register



MIN18 (#0x539, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN18 Minimum voltage (Volts) that can be set for this output.

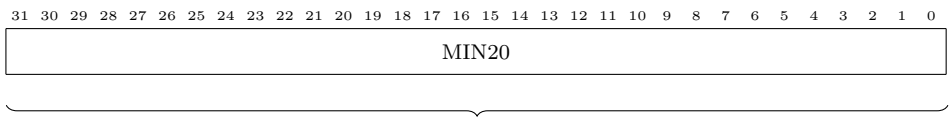
2.2.59 MIN19 (#0x53A) - OUT19 minimum voltage register



MIN19 (#0x53A, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN19 Minimum voltage (Volts) that can be set for this output.

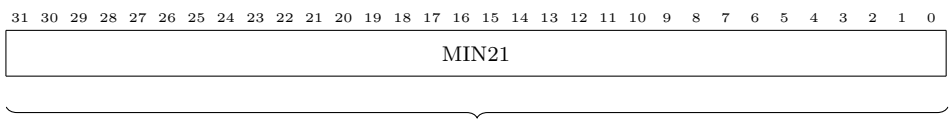
2.2.60 MIN20 (#0x53B) - OUT20 minimum voltage register



MIN20 (#0x53B, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN20 Minimum voltage (Volts) that can be set for this output.

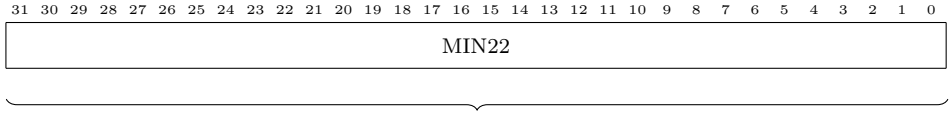
2.2.61 MIN21 (#0x53C) - OUT21 minimum voltage register



MIN21 (#0x53C, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN21 Minimum voltage (Volts) that can be set for this output.

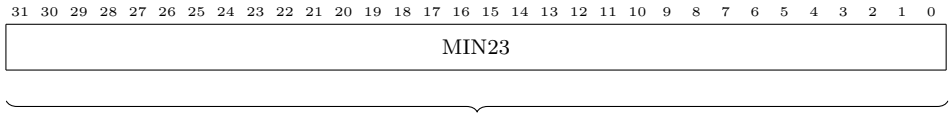
2.2.62 MIN22 (#0x53D) - OUT22 minimum voltage register



MIN22 (#0x53D, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN22 Minimum voltage (Volts) that can be set for this output.

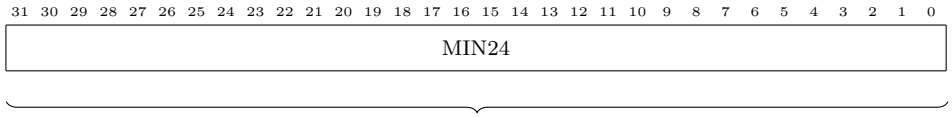
2.2.63 MIN23 (#0x53E) - OUT23 minimum voltage register



MIN23 (#0x53E, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN23 Minimum voltage (Volts) that can be set for this output.

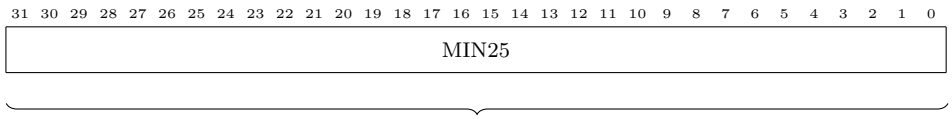
2.2.64 MIN24 (#0x53F) - OUT24 minimum voltage register



MIN24 (#0x53F, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN24 Minimum voltage (Volts) that can be set for this output.

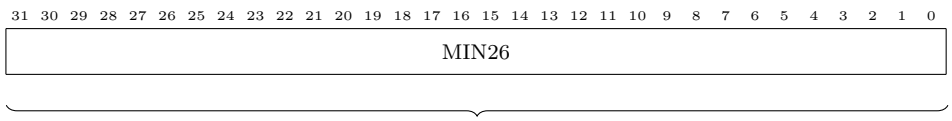
2.2.65 MIN25 (#0x540) - OUT25 minimum voltage register



MIN25 (#0x540, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN25 Minimum voltage (Volts) that can be set for this output.

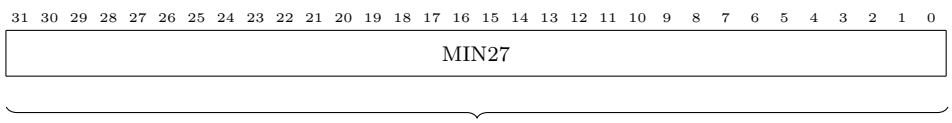
2.2.66 MIN26 (#0x541) - OUT26 minimum voltage register



MIN26 (#0x541, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN26 Minimum voltage (Volts) that can be set for this output.

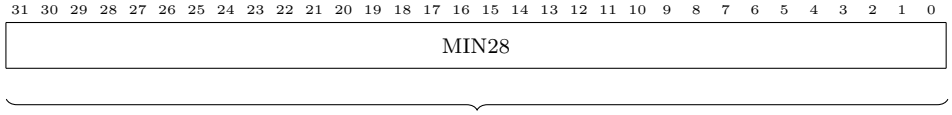
2.2.67 MIN27 (#0x542) - OUT27 minimum voltage register



MIN27 (#0x542, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN27 Minimum voltage (Volts) that can be set for this output.

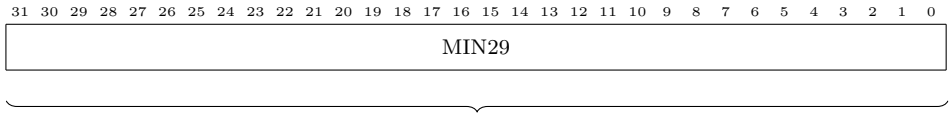
2.2.68 MIN28 (#0x543) - OUT28 minimum voltage register



MIN28 (#0x543, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN28 Minimum voltage (Volts) that can be set for this output.

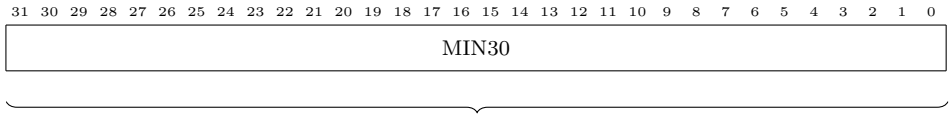
2.2.69 MIN29 (#0x544) - OUT29 minimum voltage register



MIN29 (#0x544, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN29 Minimum voltage (Volts) that can be set for this output.

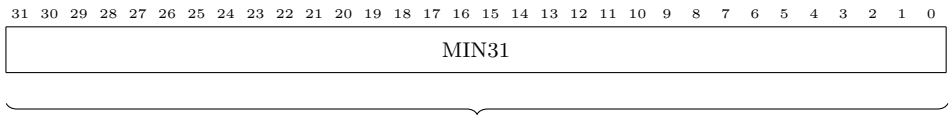
2.2.70 MIN30 (#0x545) - OUT30 minimum voltage register



MIN30 (#0x545, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN30 Minimum voltage (Volts) that can be set for this output.

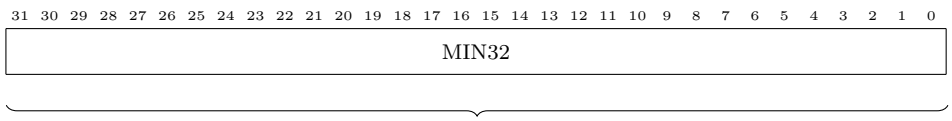
2.2.71 MIN31 (#0x546) - OUT31 minimum voltage register



MIN31 (#0x546, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN31 Minimum voltage (Volts) that can be set for this output.

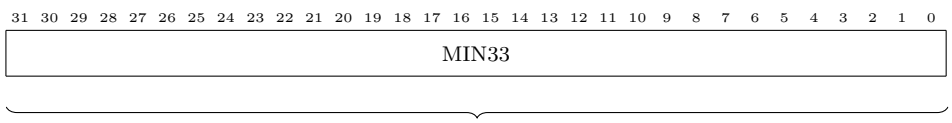
2.2.72 MIN32 (#0x547) - OUT32 minimum voltage register



MIN32 (#0x547, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN32 Minimum voltage (Volts) that can be set for this output.

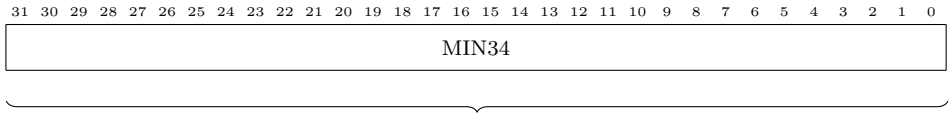
2.2.73 MIN33 (#0x548) - OUT33 minimum voltage register



MIN33 (#0x548, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN33 Minimum voltage (Volts) that can be set for this output.

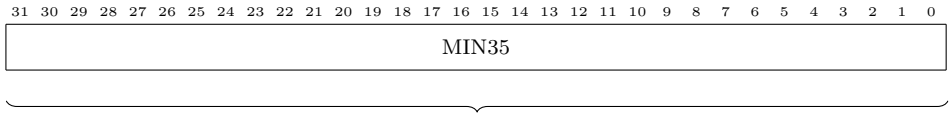
2.2.74 MIN34 (#0x549) - OUT34 minimum voltage register



MIN34 (#0x549, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN34 Minimum voltage (Volts) that can be set for this output.

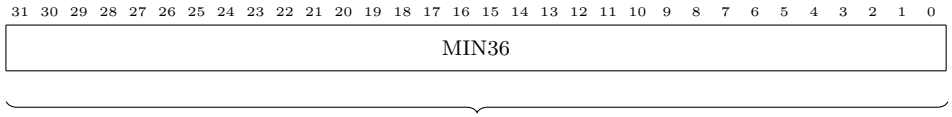
2.2.75 MIN35 (#0x54A) - OUT35 minimum voltage register



MIN35 (#0x54A, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN35 Minimum voltage (Volts) that can be set for this output.

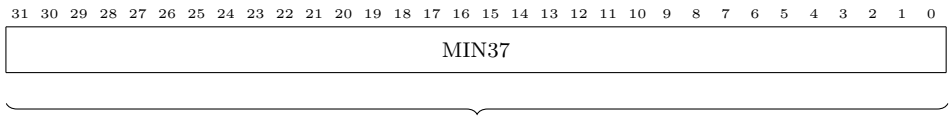
2.2.76 MIN36 (#0x54B) - OUT36 minimum voltage register



MIN36 (#0x54B, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN36 Minimum voltage (Volts) that can be set for this output.

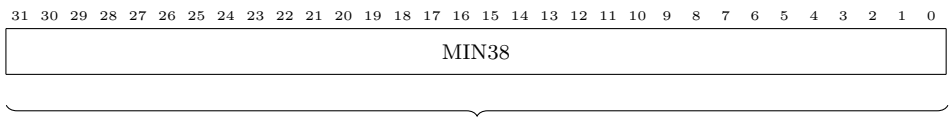
2.2.77 MIN37 (#0x54C) - OUT37 minimum voltage register



MIN37 (#0x54C, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN37 Minimum voltage (Volts) that can be set for this output.

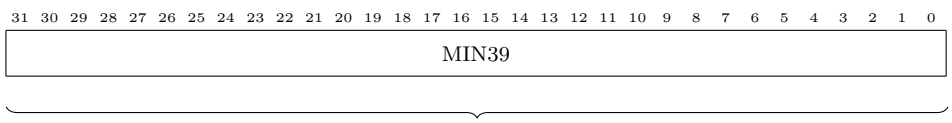
2.2.78 MIN38 (#0x54D) - OUT38 minimum voltage register



MIN38 (#0x54D, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN38 Minimum voltage (Volts) that can be set for this output.

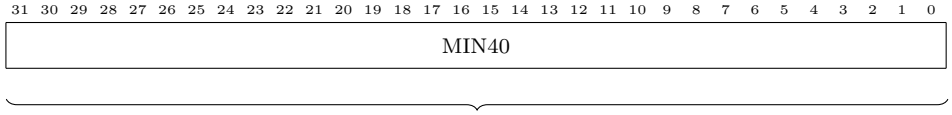
2.2.79 MIN39 (#0x54E) - OUT39 minimum voltage register



MIN39 (#0x54E, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN39 Minimum voltage (Volts) that can be set for this output.

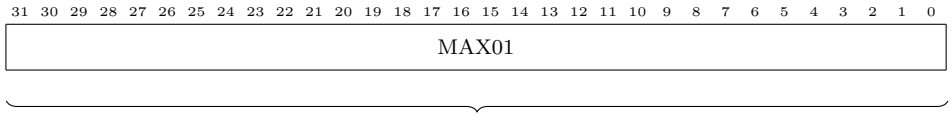
2.2.80 MIN40 (#0x54F) - OUT40 minimum voltage register



MIN40 (#0x54F, RWN, init.: 0.0, min.: 0.0, max.: 5.0)

[31:0] MIN40 Minimum voltage (Volts) that can be set for this output.

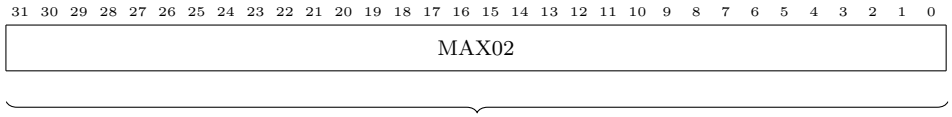
2.2.81 MAX01 (#0x550) - OUT01 maximum voltage register



MAX01 (#0x550, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX01 Maximum voltage (Volts) that can be set for this output.

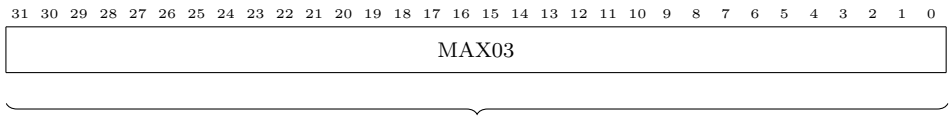
2.2.82 MAX02 (#0x551) - OUT02 maximum voltage register



MAX02 (#0x551, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX02 Maximum voltage (Volts) that can be set for this output.

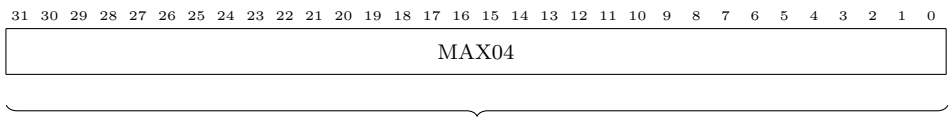
2.2.83 MAX03 (#0x552) - OUT03 maximum voltage register



MAX03 (#0x552, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX03 Maximum voltage (Volts) that can be set for this output.

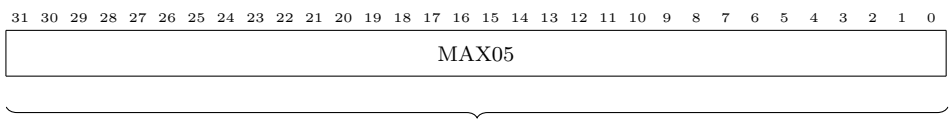
2.2.84 MAX04 (#0x553) - OUT04 maximum voltage register



MAX04 (#0x553, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX04 Maximum voltage (Volts) that can be set for this output.

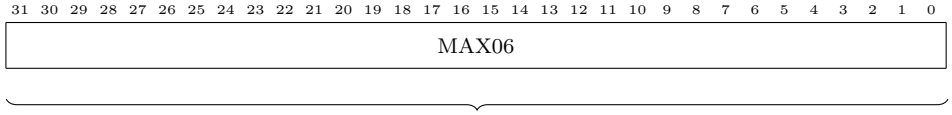
2.2.85 MAX05 (#0x554) - OUT05 maximum voltage register



MAX05 (#0x554, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX05 Maximum voltage (Volts) that can be set for this output.

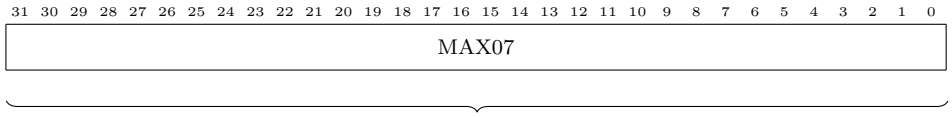
2.2.86 MAX06 (#0x555) - OUT06 maximum voltage register



MAX06 (#0x555, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX06 Maximum voltage (Volts) that can be set for this output.

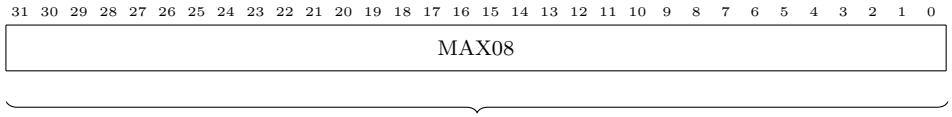
2.2.87 MAX07 (#0x556) - OUT07 maximum voltage register



MAX07 (#0x556, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX07 Maximum voltage (Volts) that can be set for this output.

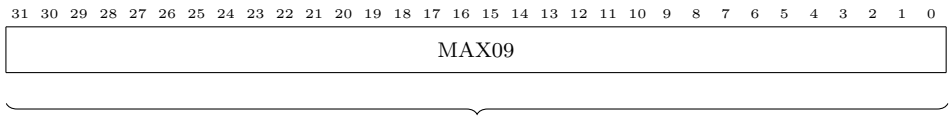
2.2.88 MAX08 (#0x557) - OUT08 maximum voltage register



MAX08 (#0x557, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX08 Maximum voltage (Volts) that can be set for this output.

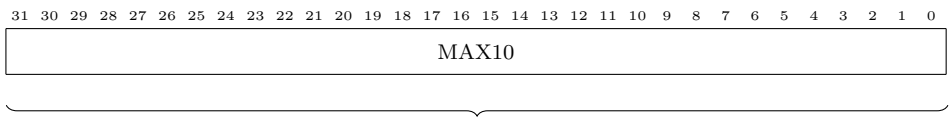
2.2.89 MAX09 (#0x558) - OUT09 maximum voltage register



MAX09 (#0x558, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX09 Maximum voltage (Volts) that can be set for this output.

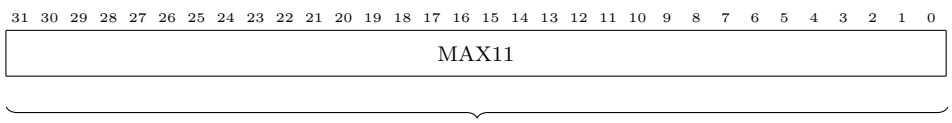
2.2.90 MAX10 (#0x559) - OUT10 maximum voltage register



MAX10 (#0x559, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX10 Maximum voltage (Volts) that can be set for this output.

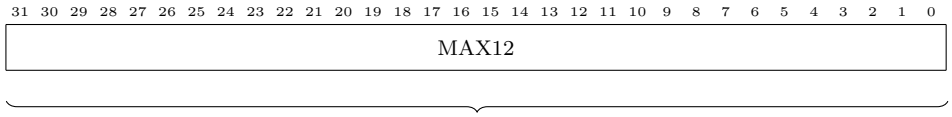
2.2.91 MAX11 (#0x55A) - OUT11 maximum voltage register



MAX11 (#0x55A, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX11 Maximum voltage (Volts) that can be set for this output.

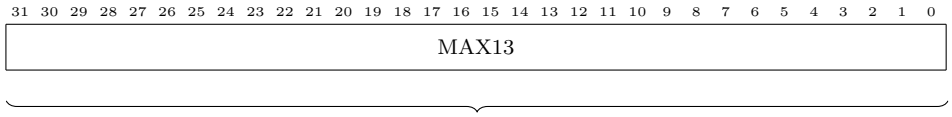
2.2.92 MAX12 (#0x55B) - OUT12 maximum voltage register



MAX12 (#0x55B, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX12 Maximum voltage (Volts) that can be set for this output.

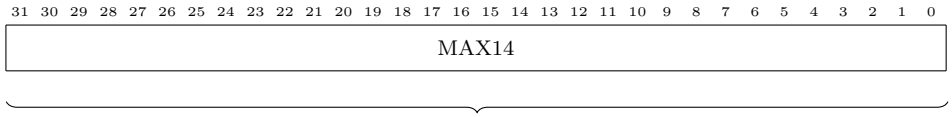
2.2.93 MAX13 (#0x55C) - OUT13 maximum voltage register



MAX13 (#0x55C, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX13 Maximum voltage (Volts) that can be set for this output.

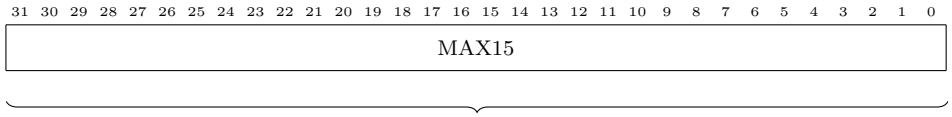
2.2.94 MAX14 (#0x55D) - OUT14 maximum voltage register



MAX14 (#0x55D, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX14 Maximum voltage (Volts) that can be set for this output.

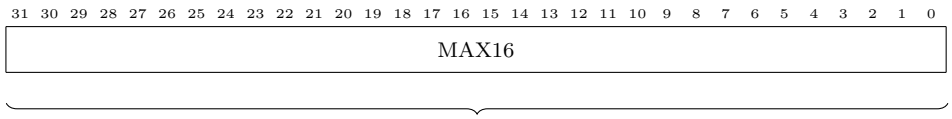
2.2.95 MAX15 (#0x55E) - OUT15 maximum voltage register



MAX15 (#0x55E, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX15 Maximum voltage (Volts) that can be set for this output.

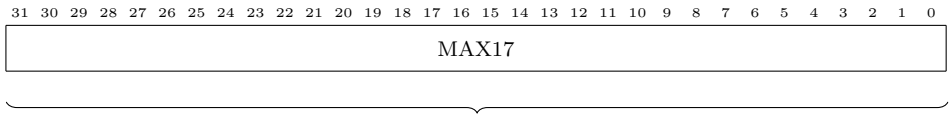
2.2.96 MAX16 (#0x55F) - OUT16 maximum voltage register



MAX16 (#0x55F, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX16 Maximum voltage (Volts) that can be set for this output.

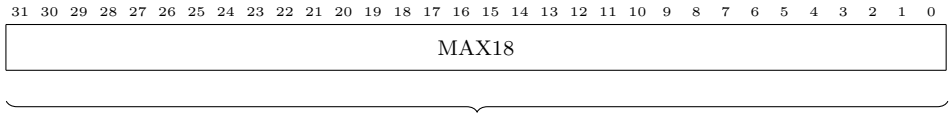
2.2.97 MAX17 (#0x560) - OUT17 maximum voltage register



MAX17 (#0x560, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX17 Maximum voltage (Volts) that can be set for this output.

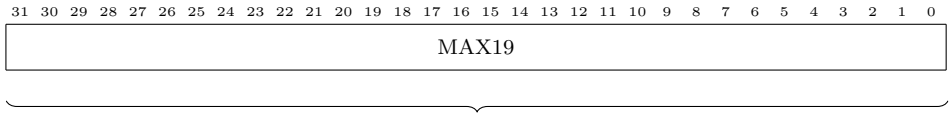
2.2.98 MAX18 (#0x561) - OUT18 maximum voltage register



MAX18 (#0x561, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX18 Maximum voltage (Volts) that can be set for this output.

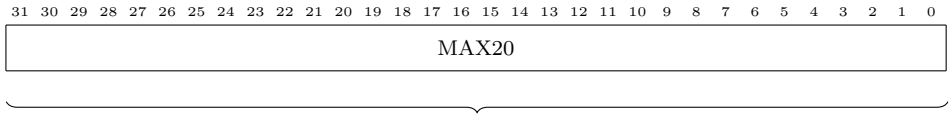
2.2.99 MAX19 (#0x562) - OUT19 maximum voltage register



MAX19 (#0x562, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX19 Maximum voltage (Volts) that can be set for this output.

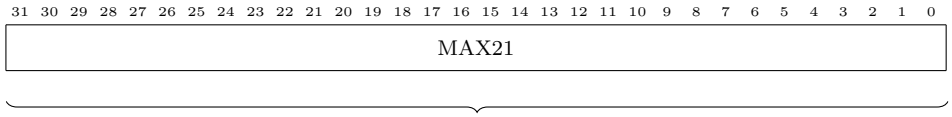
2.2.100 MAX20 (#0x563) - OUT20 maximum voltage register



MAX20 (#0x563, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX20 Maximum voltage (Volts) that can be set for this output.

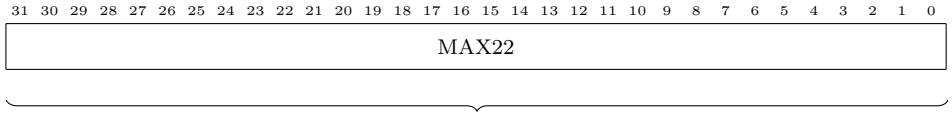
2.2.101 MAX21 (#0x564) - OUT21 maximum voltage register



MAX21 (#0x564, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX21 Maximum voltage (Volts) that can be set for this output.

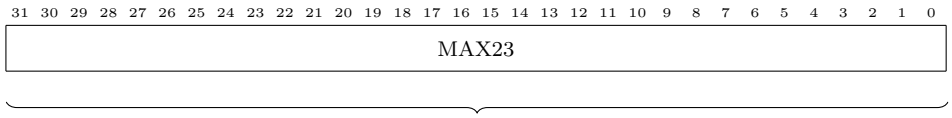
2.2.102 MAX22 (#0x565) - OUT22 maximum voltage register



MAX22 (#0x565, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX22 Maximum voltage (Volts) that can be set for this output.

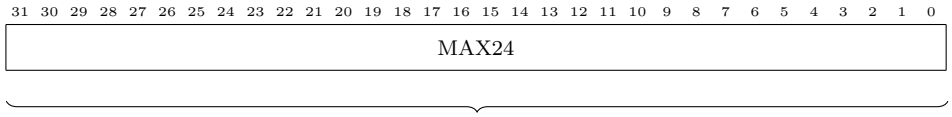
2.2.103 MAX23 (#0x566) - OUT23 maximum voltage register



MAX23 (#0x566, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX23 Maximum voltage (Volts) that can be set for this output.

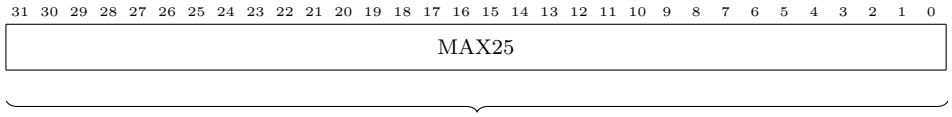
2.2.104 MAX24 (#0x567) - OUT24 maximum voltage register



MAX24 (#0x567, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX24 Maximum voltage (Volts) that can be set for this output.

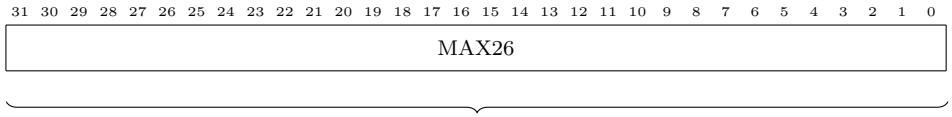
2.2.105 MAX25 (#0x568) - OUT25 maximum voltage register



MAX25 (#0x568, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX25 Maximum voltage (Volts) that can be set for this output.

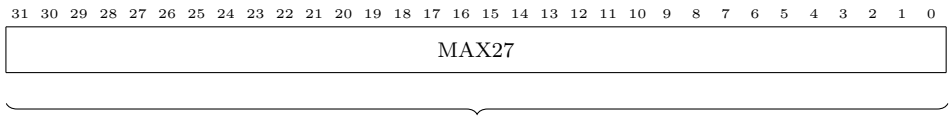
2.2.106 MAX26 (#0x569) - OUT26 maximum voltage register



MAX26 (#0x569, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX26 Maximum voltage (Volts) that can be set for this output.

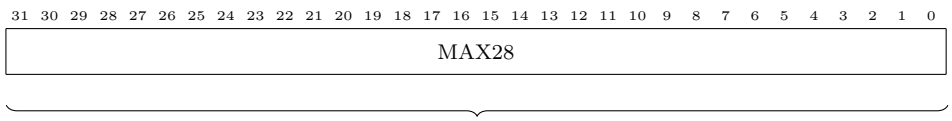
2.2.107 MAX27 (#0x56A) - OUT27 maximum voltage register



MAX27 (#0x56A, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX27 Maximum voltage (Volts) that can be set for this output.

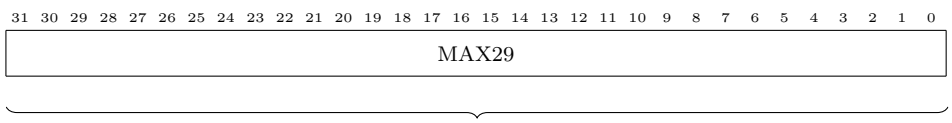
2.2.108 MAX28 (#0x56B) - OUT28 maximum voltage register



MAX28 (#0x56B, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX28 Maximum voltage (Volts) that can be set for this output.

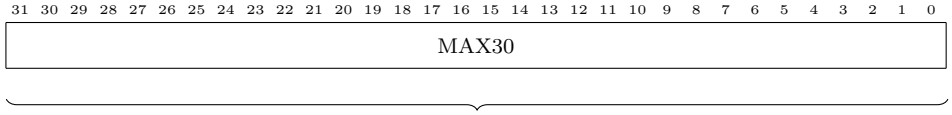
2.2.109 MAX29 (#0x56C) - OUT29 maximum voltage register



MAX29 (#0x56C, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX29 Maximum voltage (Volts) that can be set for this output.

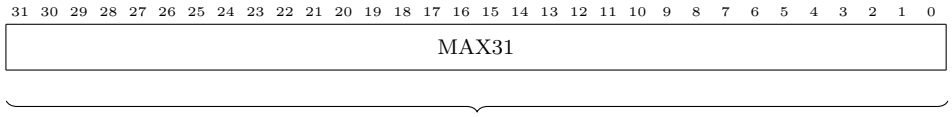
2.2.110 MAX30 (#0x56D) - OUT30 maximum voltage register



MAX30 (#0x56D, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX30 Maximum voltage (Volts) that can be set for this output.

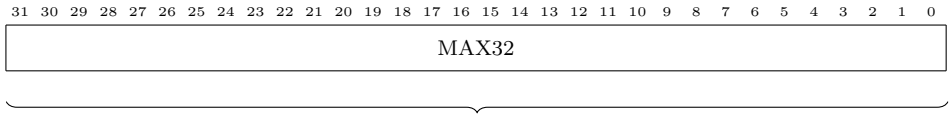
2.2.111 MAX31 (#0x56E) - OUT31 maximum voltage register



MAX31 (#0x56E, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX31 Maximum voltage (Volts) that can be set for this output.

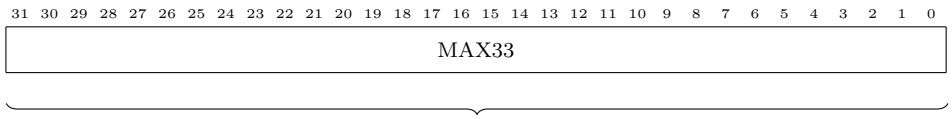
2.2.112 MAX32 (#0x56F) - OUT32 maximum voltage register



MAX32 (#0x56F, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX32 Maximum voltage (Volts) that can be set for this output.

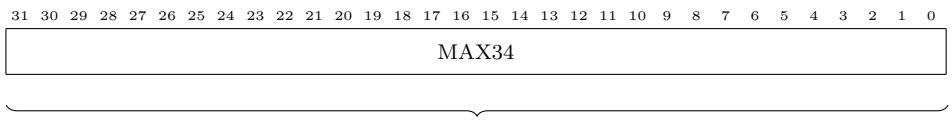
2.2.113 MAX33 (#0x570) - OUT33 maximum voltage register



MAX33 (#0x570, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX33 Maximum voltage (Volts) that can be set for this output.

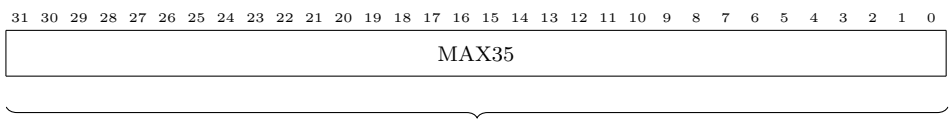
2.2.114 MAX34 (#0x571) - OUT34 maximum voltage register



MAX34 (#0x571, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX34 Maximum voltage (Volts) that can be set for this output.

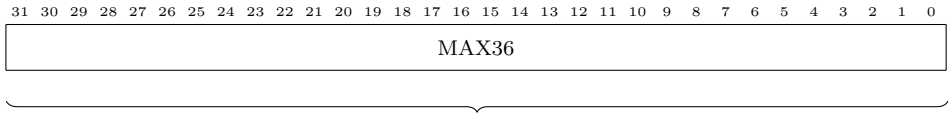
2.2.115 MAX35 (#0x572) - OUT35 maximum voltage register



MAX35 (#0x572, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX35 Maximum voltage (Volts) that can be set for this output.

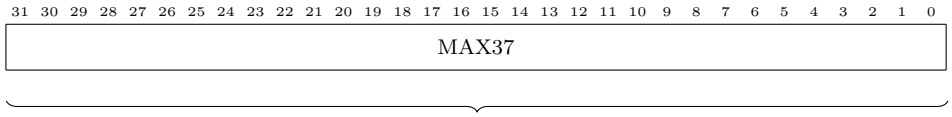
2.2.116 MAX36 (#0x573) - OUT36 maximum voltage register



MAX36 (#0x573, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX36 Maximum voltage (Volts) that can be set for this output.

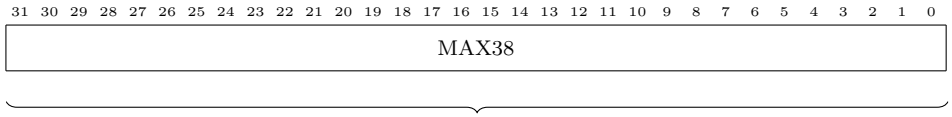
2.2.117 MAX37 (#0x574) - OUT37 maximum voltage register



MAX37 (#0x574, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX37 Maximum voltage (Volts) that can be set for this output.

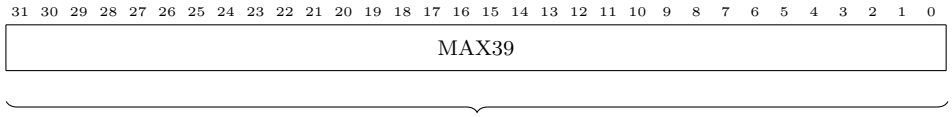
2.2.118 MAX38 (#0x575) - OUT38 maximum voltage register



MAX38 (#0x575, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX38 Maximum voltage (Volts) that can be set for this output.

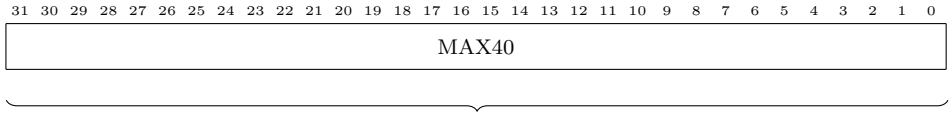
2.2.119 MAX39 (#0x576) - OUT39 maximum voltage register



MAX39 (#0x576, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX39 Maximum voltage (Volts) that can be set for this output.

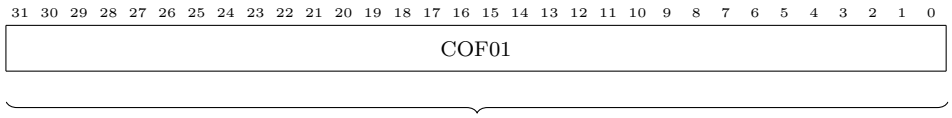
2.2.120 MAX40 (#0x577) - OUT40 maximum voltage register



MAX40 (#0x577, RWN, init.: 3.0, min.: 0.0, max.: 5.0)

[31:0] MAX40 Maximum voltage (Volts) that can be set for this output.

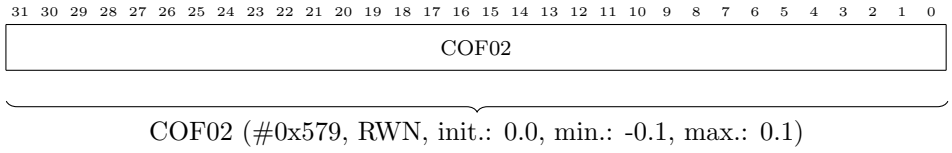
2.2.121 COF01 (#0x578) - OUT01 calibration offset register



COF01 (#0x578, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF01 Calibration offset (Volts), see DAC calibration.

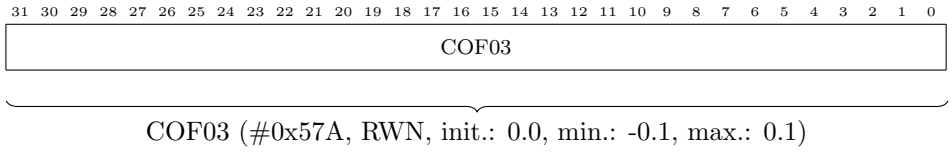
2.2.122 COF02 (#0x579) - OUT02 calibration offset register



COF02 (#0x579, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF02 Calibration offset (Volts), see DAC calibration.

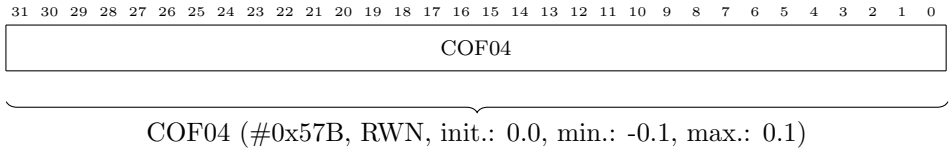
2.2.123 COF03 (#0x57A) - OUT03 calibration offset register



COF03 (#0x57A, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF03 Calibration offset (Volts), see DAC calibration.

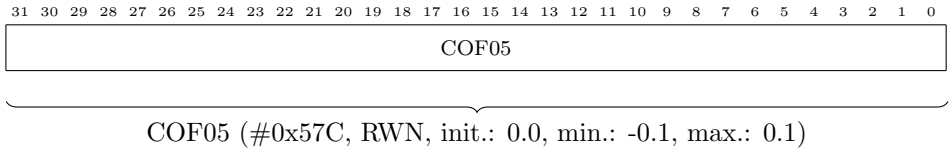
2.2.124 COF04 (#0x57B) - OUT04 calibration offset register



COF04 (#0x57B, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF04 Calibration offset (Volts), see DAC calibration.

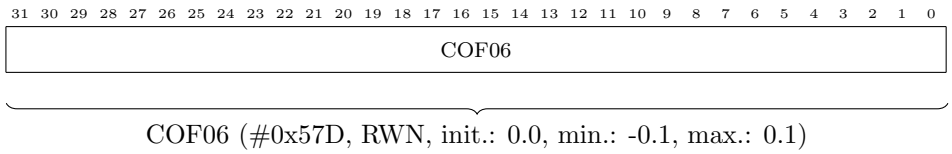
2.2.125 COF05 (#0x57C) - OUT05 calibration offset register



COF05 (#0x57C, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF05 Calibration offset (Volts), see DAC calibration.

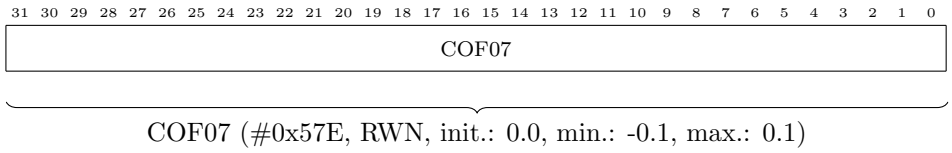
2.2.126 COF06 (#0x57D) - OUT06 calibration offset register



COF06 (#0x57D, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF06 Calibration offset (Volts), see DAC calibration.

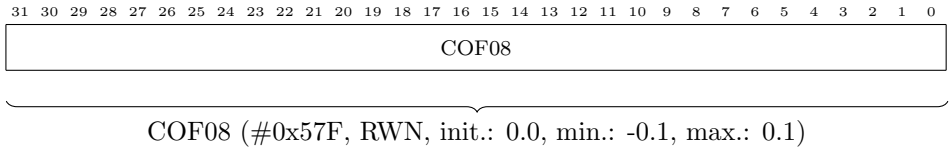
2.2.127 COF07 (#0x57E) - OUT07 calibration offset register



COF07 (#0x57E, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF07 Calibration offset (Volts), see DAC calibration.

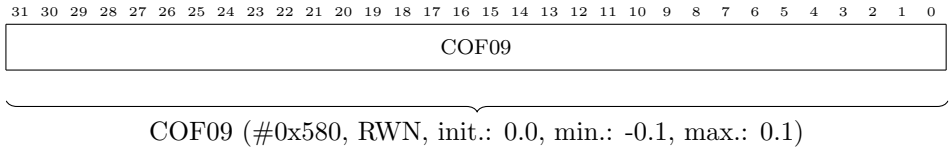
2.2.128 COF08 (#0x57F) - OUT08 calibration offset register



COF08 (#0x57F, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF08 Calibration offset (Volts), see DAC calibration.

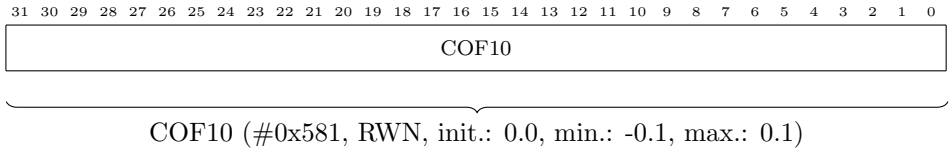
2.2.129 COF09 (#0x580) - OUT09 calibration offset register



COF09 (#0x580, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF09 Calibration offset (Volts), see DAC calibration.

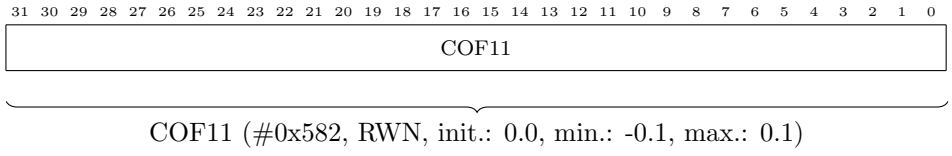
2.2.130 COF10 (#0x581) - OUT10 calibration offset register



COF10 (#0x581, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF10 Calibration offset (Volts), see DAC calibration.

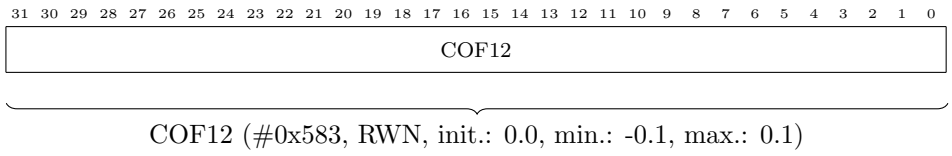
2.2.131 COF11 (#0x582) - OUT11 calibration offset register



COF11 (#0x582, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF11 Calibration offset (Volts), see DAC calibration.

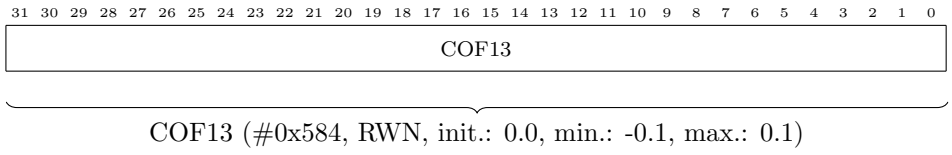
2.2.132 COF12 (#0x583) - OUT12 calibration offset register



COF12 (#0x583, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF12 Calibration offset (Volts), see DAC calibration.

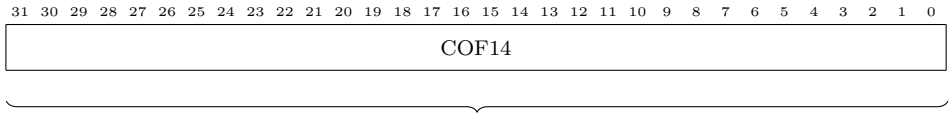
2.2.133 COF13 (#0x584) - OUT13 calibration offset register



COF13 (#0x584, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF13 Calibration offset (Volts), see DAC calibration.

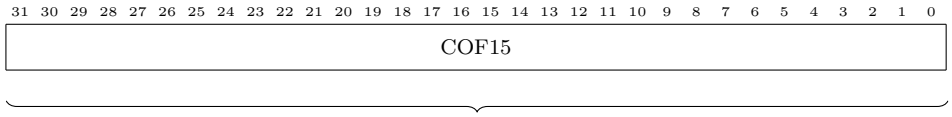
2.2.134 COF14 (#0x585) - OUT14 calibration offset register



COF14 (#0x585, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF14 Calibration offset (Volts), see DAC calibration.

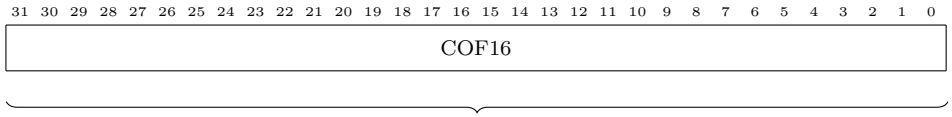
2.2.135 COF15 (#0x586) - OUT15 calibration offset register



COF15 (#0x586, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF15 Calibration offset (Volts), see DAC calibration.

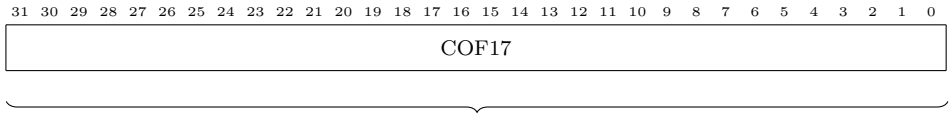
2.2.136 COF16 (#0x587) - OUT16 calibration offset register



COF16 (#0x587, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF16 Calibration offset (Volts), see DAC calibration.

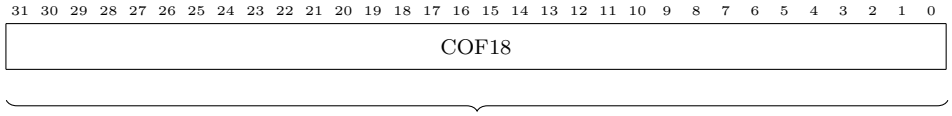
2.2.137 COF17 (#0x588) - OUT17 calibration offset register



COF17 (#0x588, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF17 Calibration offset (Volts), see DAC calibration.

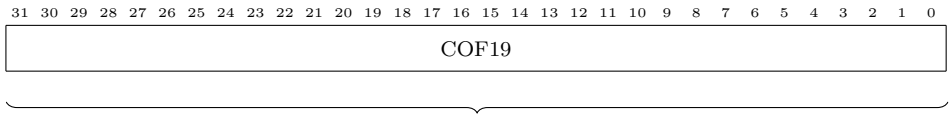
2.2.138 COF18 (#0x589) - OUT18 calibration offset register



COF18 (#0x589, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF18 Calibration offset (Volts), see DAC calibration.

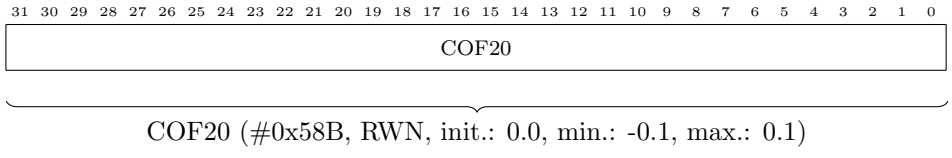
2.2.139 COF19 (#0x58A) - OUT19 calibration offset register



COF19 (#0x58A, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF19 Calibration offset (Volts), see DAC calibration.

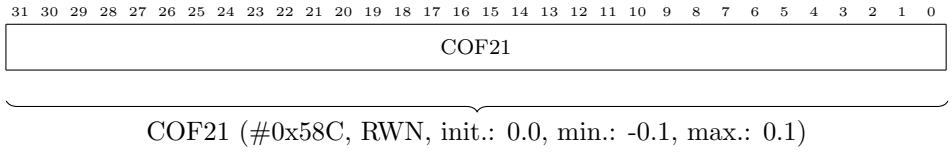
2.2.140 COF20 (#0x58B) - OUT20 calibration offset register



COF20 (#0x58B, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF20 Calibration offset (Volts), see DAC calibration.

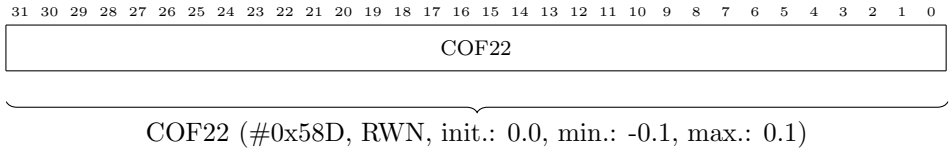
2.2.141 COF21 (#0x58C) - OUT21 calibration offset register



COF21 (#0x58C, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF21 Calibration offset (Volts), see DAC calibration.

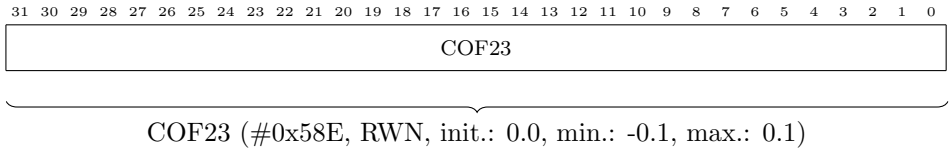
2.2.142 COF22 (#0x58D) - OUT22 calibration offset register



COF22 (#0x58D, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF22 Calibration offset (Volts), see DAC calibration.

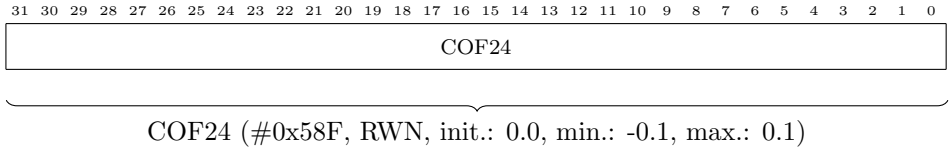
2.2.143 COF23 (#0x58E) - OUT23 calibration offset register



COF23 (#0x58E, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF23 Calibration offset (Volts), see DAC calibration.

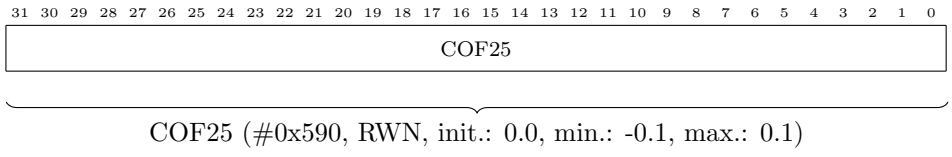
2.2.144 COF24 (#0x58F) - OUT24 calibration offset register



COF24 (#0x58F, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF24 Calibration offset (Volts), see DAC calibration.

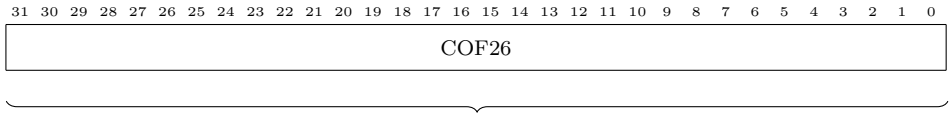
2.2.145 COF25 (#0x590) - OUT25 calibration offset register



COF25 (#0x590, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF25 Calibration offset (Volts), see DAC calibration.

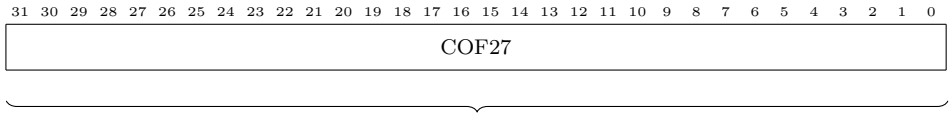
2.2.146 COF26 (#0x591) - OUT26 calibration offset register



COF26 (#0x591, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF26 Calibration offset (Volts), see DAC calibration.

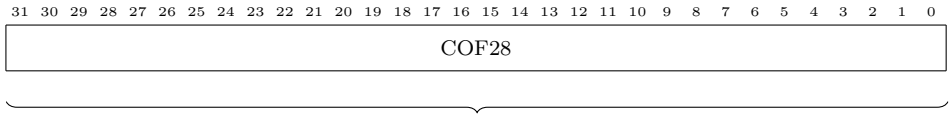
2.2.147 COF27 (#0x592) - OUT27 calibration offset register



COF27 (#0x592, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF27 Calibration offset (Volts), see DAC calibration.

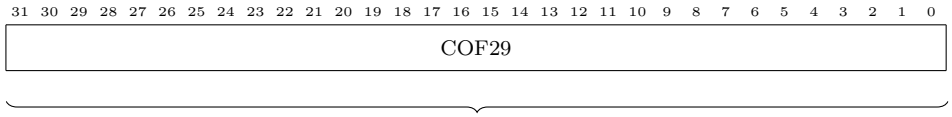
2.2.148 COF28 (#0x593) - OUT28 calibration offset register



COF28 (#0x593, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF28 Calibration offset (Volts), see DAC calibration.

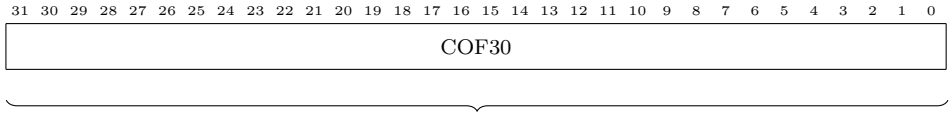
2.2.149 COF29 (#0x594) - OUT29 calibration offset register



COF29 (#0x594, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF29 Calibration offset (Volts), see DAC calibration.

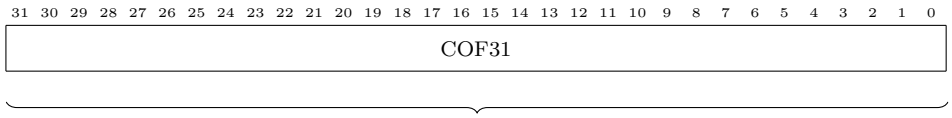
2.2.150 COF30 (#0x595) - OUT30 calibration offset register



COF30 (#0x595, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF30 Calibration offset (Volts), see DAC calibration.

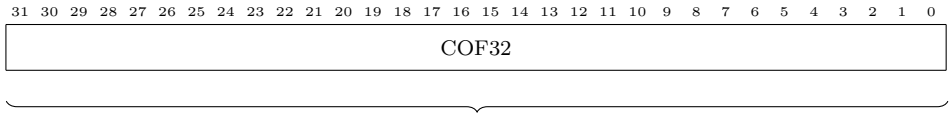
2.2.151 COF31 (#0x596) - OUT31 calibration offset register



COF31 (#0x596, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF31 Calibration offset (Volts), see DAC calibration.

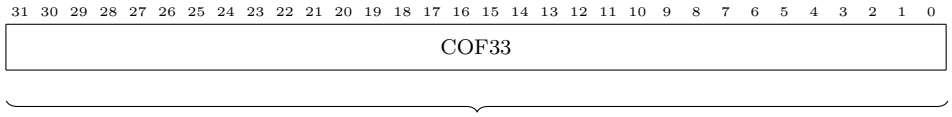
2.2.152 COF32 (#0x597) - OUT32 calibration offset register



COF32 (#0x597, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF32 Calibration offset (Volts), see DAC calibration.

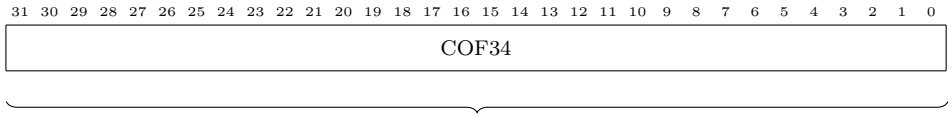
2.2.153 COF33 (#0x598) - OUT33 calibration offset register



COF33 (#0x598, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF33 Calibration offset (Volts), see DAC calibration.

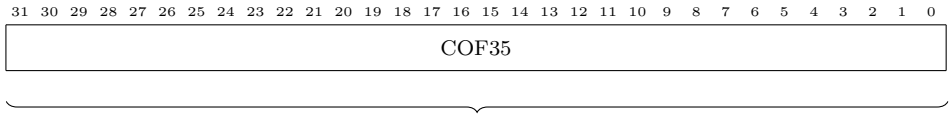
2.2.154 COF34 (#0x599) - OUT34 calibration offset register



COF34 (#0x599, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF34 Calibration offset (Volts), see DAC calibration.

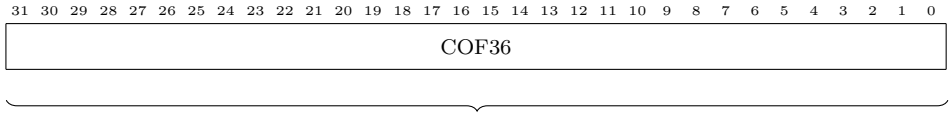
2.2.155 COF35 (#0x59A) - OUT35 calibration offset register



COF35 (#0x59A, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF35 Calibration offset (Volts), see DAC calibration.

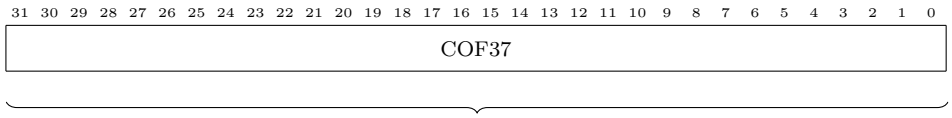
2.2.156 COF36 (#0x59B) - OUT36 calibration offset register



COF36 (#0x59B, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF36 Calibration offset (Volts), see DAC calibration.

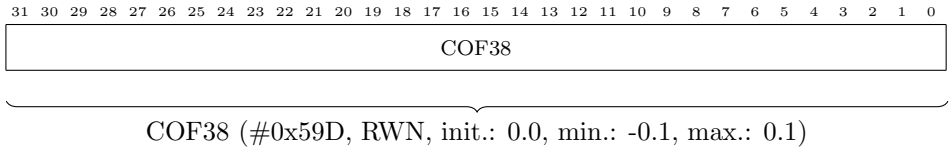
2.2.157 COF37 (#0x59C) - OUT37 calibration offset register



COF37 (#0x59C, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF37 Calibration offset (Volts), see DAC calibration.

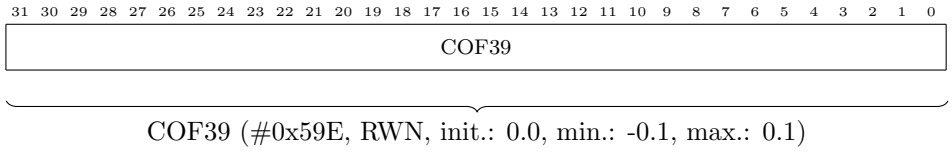
2.2.158 COF38 (#0x59D) - OUT38 calibration offset register



COF38 (#0x59D, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF38 Calibration offset (Volts), see DAC calibration.

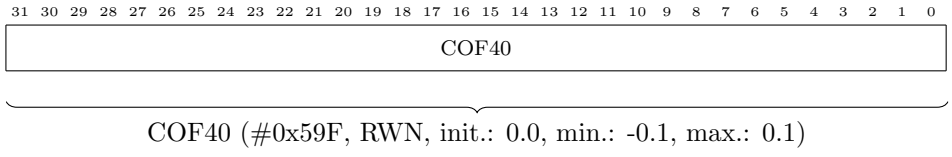
2.2.159 COF39 (#0x59E) - OUT39 calibration offset register



COF39 (#0x59E, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF39 Calibration offset (Volts), see DAC calibration.

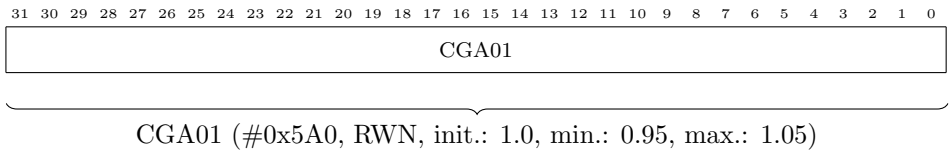
2.2.160 COF40 (#0x59F) - OUT40 calibration offset register



COF40 (#0x59F, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF40 Calibration offset (Volts), see DAC calibration.

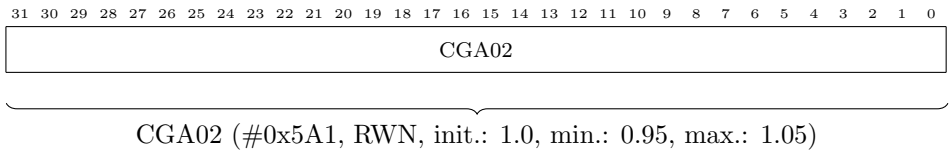
2.2.161 CGA01 (#0x5A0) - OUT01 calibration gain register



CGA01 (#0x5A0, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA01 Calibration gain (Volts/Volts), see DAC calibration.

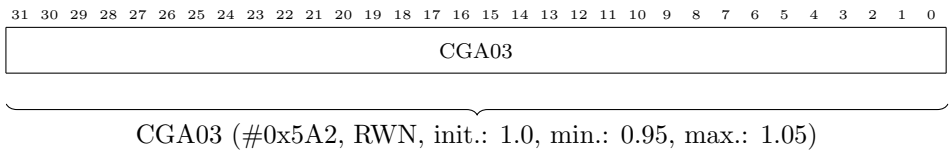
2.2.162 CGA02 (#0x5A1) - OUT02 calibration gain register



CGA02 (#0x5A1, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA02 Calibration gain (Volts/Volts), see DAC calibration.

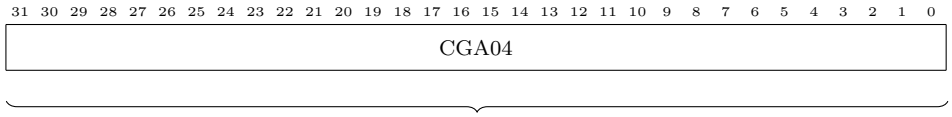
2.2.163 CGA03 (#0x5A2) - OUT03 calibration gain register



CGA03 (#0x5A2, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA03 Calibration gain (Volts/Volts), see DAC calibration.

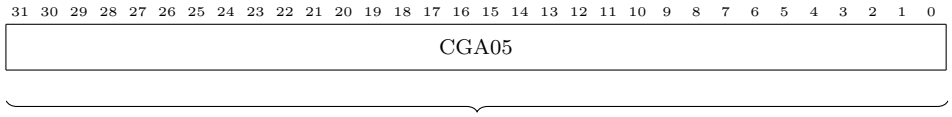
2.2.164 CGA04 (#0x5A3) - OUT04 calibration gain register



CGA04 (#0x5A3, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA04 Calibration gain (Volts/Volts), see DAC calibration.

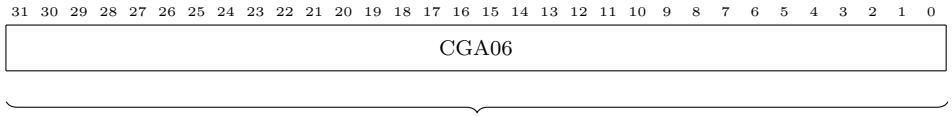
2.2.165 CGA05 (#0x5A4) - OUT05 calibration gain register



CGA05 (#0x5A4, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA05 Calibration gain (Volts/Volts), see DAC calibration.

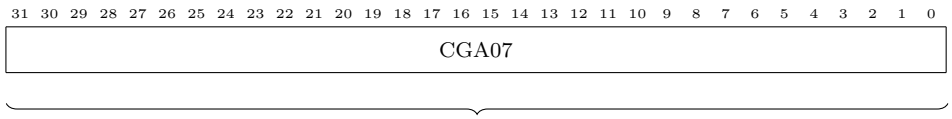
2.2.166 CGA06 (#0x5A5) - OUT06 calibration gain register



CGA06 (#0x5A5, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA06 Calibration gain (Volts/Volts), see DAC calibration.

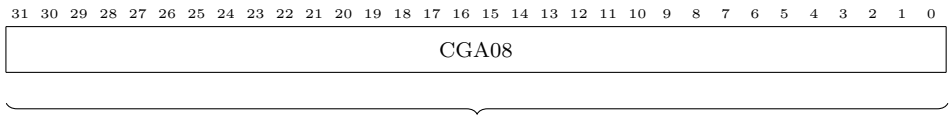
2.2.167 CGA07 (#0x5A6) - OUT07 calibration gain register



CGA07 (#0x5A6, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA07 Calibration gain (Volts/Volts), see DAC calibration.

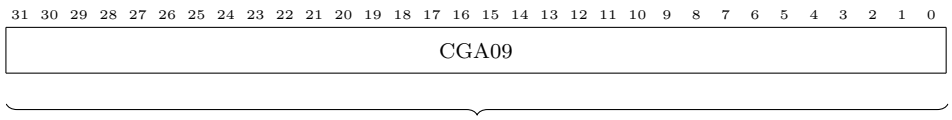
2.2.168 CGA08 (#0x5A7) - OUT08 calibration gain register



CGA08 (#0x5A7, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA08 Calibration gain (Volts/Volts), see DAC calibration.

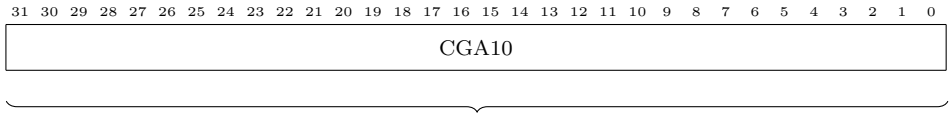
2.2.169 CGA09 (#0x5A8) - OUT09 calibration gain register



CGA09 (#0x5A8, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA09 Calibration gain (Volts/Volts), see DAC calibration.

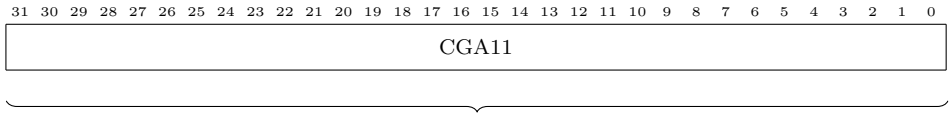
2.2.170 CGA10 (#0x5A9) - OUT10 calibration gain register



CGA10 (#0x5A9, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA10 Calibration gain (Volts/Volts), see DAC calibration.

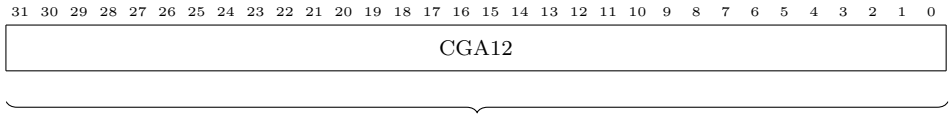
2.2.171 CGA11 (#0x5AA) - OUT11 calibration gain register



CGA11 (#0x5AA, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA11 Calibration gain (Volts/Volts), see DAC calibration.

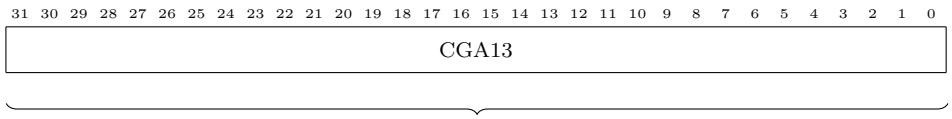
2.2.172 CGA12 (#0x5AB) - OUT12 calibration gain register



CGA12 (#0x5AB, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA12 Calibration gain (Volts/Volts), see DAC calibration.

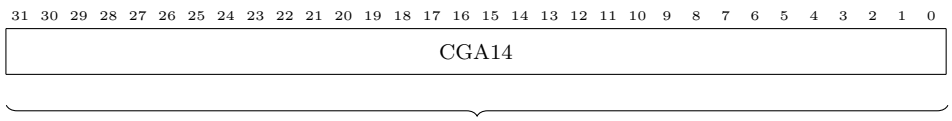
2.2.173 CGA13 (#0x5AC) - OUT13 calibration gain register



CGA13 (#0x5AC, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA13 Calibration gain (Volts/Volts), see DAC calibration.

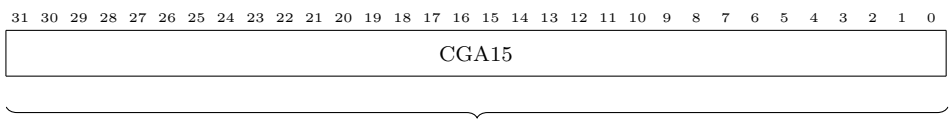
2.2.174 CGA14 (#0x5AD) - OUT14 calibration gain register



CGA14 (#0x5AD, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA14 Calibration gain (Volts/Volts), see DAC calibration.

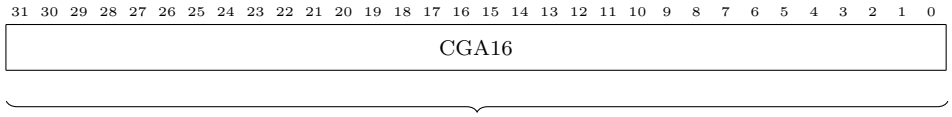
2.2.175 CGA15 (#0x5AE) - OUT15 calibration gain register



CGA15 (#0x5AE, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA15 Calibration gain (Volts/Volts), see DAC calibration.

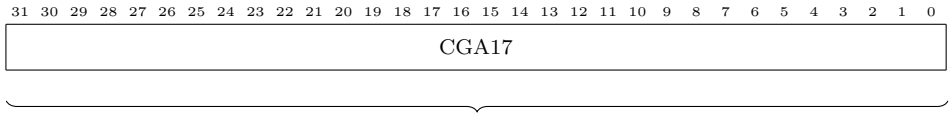
2.2.176 CGA16 (#0x5AF) - OUT16 calibration gain register



CGA16 (#0x5AF, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA16 Calibration gain (Volts/Volts), see DAC calibration.

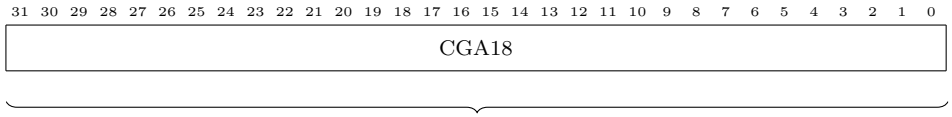
2.2.177 CGA17 (#0x5B0) - OUT17 calibration gain register



CGA17 (#0x5B0, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA17 Calibration gain (Volts/Volts), see DAC calibration.

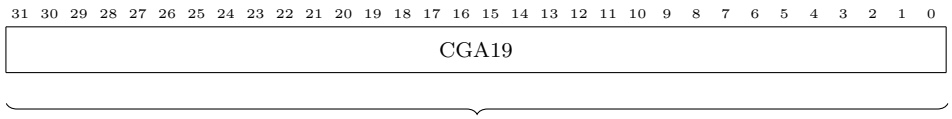
2.2.178 CGA18 (#0x5B1) - OUT18 calibration gain register



CGA18 (#0x5B1, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA18 Calibration gain (Volts/Volts), see DAC calibration.

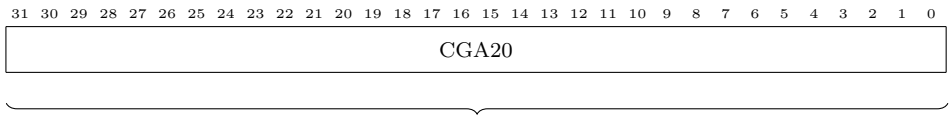
2.2.179 CGA19 (#0x5B2) - OUT19 calibration gain register



CGA19 (#0x5B2, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA19 Calibration gain (Volts/Volts), see DAC calibration.

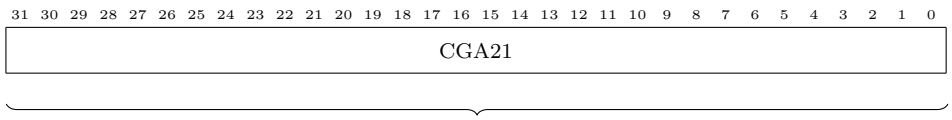
2.2.180 CGA20 (#0x5B3) - OUT20 calibration gain register



CGA20 (#0x5B3, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA20 Calibration gain (Volts/Volts), see DAC calibration.

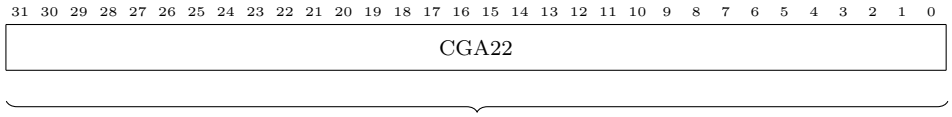
2.2.181 CGA21 (#0x5B4) - OUT21 calibration gain register



CGA21 (#0x5B4, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA21 Calibration gain (Volts/Volts), see DAC calibration.

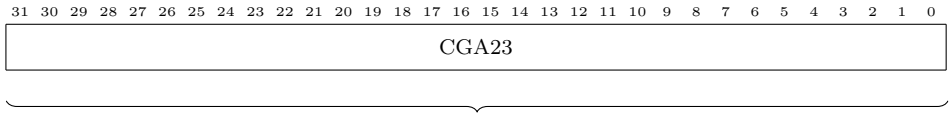
2.2.182 CGA22 (#0x5B5) - OUT22 calibration gain register



CGA22 (#0x5B5, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA22 Calibration gain (Volts/Volts), see DAC calibration.

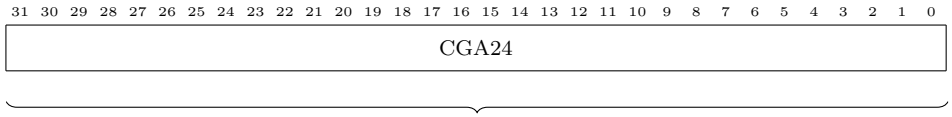
2.2.183 CGA23 (#0x5B6) - OUT23 calibration gain register



CGA23 (#0x5B6, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA23 Calibration gain (Volts/Volts), see DAC calibration.

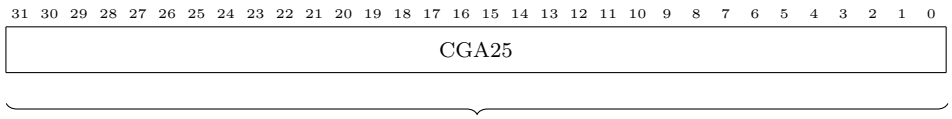
2.2.184 CGA24 (#0x5B7) - OUT24 calibration gain register



CGA24 (#0x5B7, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA24 Calibration gain (Volts/Volts), see DAC calibration.

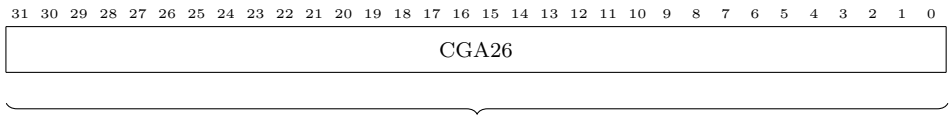
2.2.185 CGA25 (#0x5B8) - OUT25 calibration gain register



CGA25 (#0x5B8, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA25 Calibration gain (Volts/Volts), see DAC calibration.

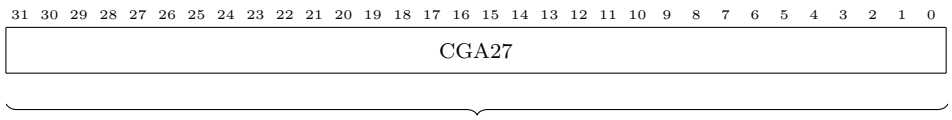
2.2.186 CGA26 (#0x5B9) - OUT26 calibration gain register



CGA26 (#0x5B9, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA26 Calibration gain (Volts/Volts), see DAC calibration.

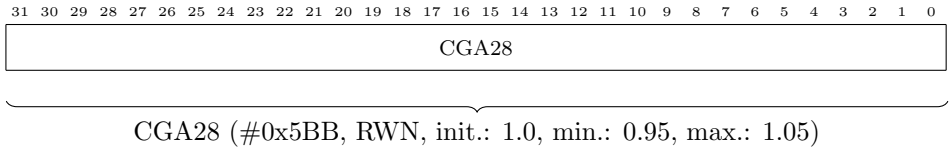
2.2.187 CGA27 (#0x5BA) - OUT27 calibration gain register



CGA27 (#0x5BA, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

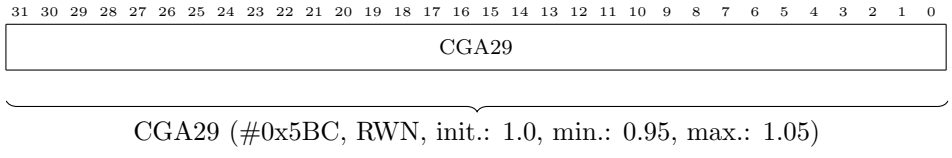
[31:0] CGA27 Calibration gain (Volts/Volts), see DAC calibration.

2.2.188 CGA28 (#0x5BB) - OUT28 calibration gain register



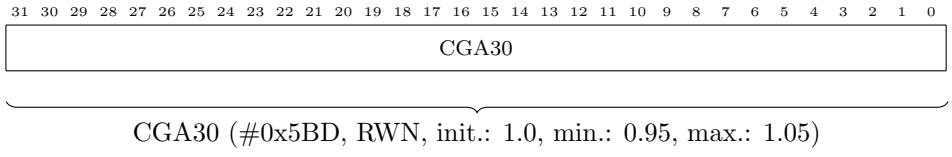
[31:0] CGA28 Calibration gain (Volts/Volts), see DAC calibration.

2.2.189 CGA29 (#0x5BC) - OUT29 calibration gain register



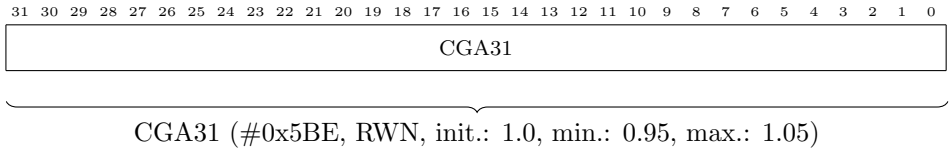
[31:0] CGA29 Calibration gain (Volts/Volts), see DAC calibration.

2.2.190 CGA30 (#0x5BD) - OUT30 calibration gain register



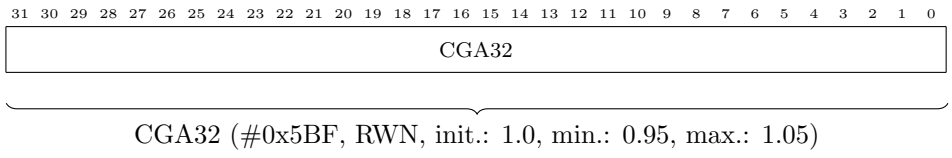
[31:0] CGA30 Calibration gain (Volts/Volts), see DAC calibration.

2.2.191 CGA31 (#0x5BE) - OUT31 calibration gain register



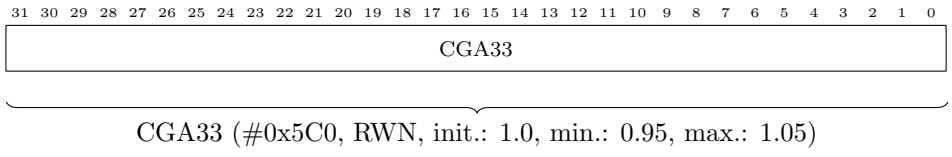
[31:0] CGA31 Calibration gain (Volts/Volts), see DAC calibration.

2.2.192 CGA32 (#0x5BF) - OUT32 calibration gain register



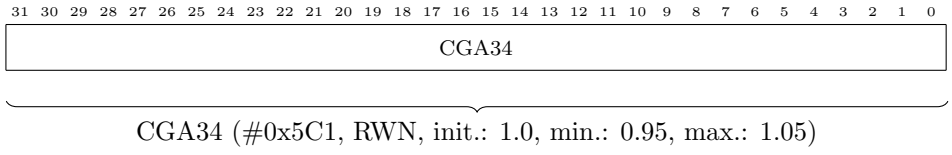
[31:0] CGA32 Calibration gain (Volts/Volts), see DAC calibration.

2.2.193 CGA33 (#0x5C0) - OUT33 calibration gain register



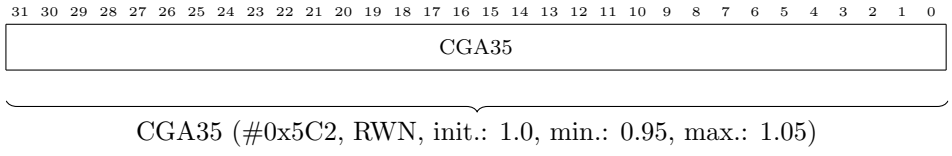
[31:0] CGA33 Calibration gain (Volts/Volts), see DAC calibration.

2.2.194 CGA34 (#0x5C1) - OUT34 calibration gain register



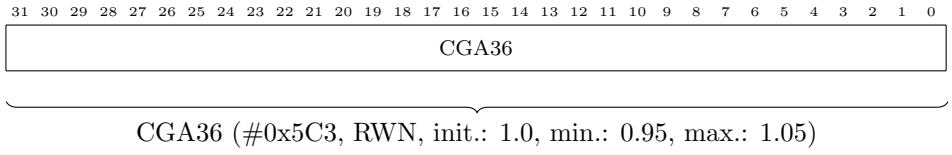
[31:0] CGA34 Calibration gain (Volts/Volts), see DAC calibration.

2.2.195 CGA35 (#0x5C2) - OUT35 calibration gain register



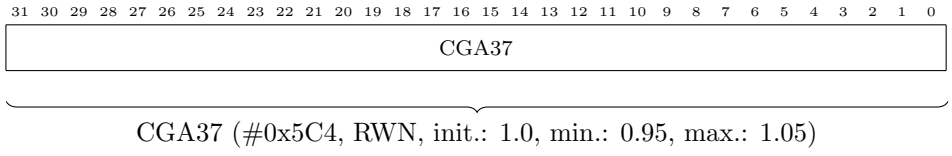
[31:0] CGA35 Calibration gain (Volts/Volts), see DAC calibration.

2.2.196 CGA36 (#0x5C3) - OUT36 calibration gain register



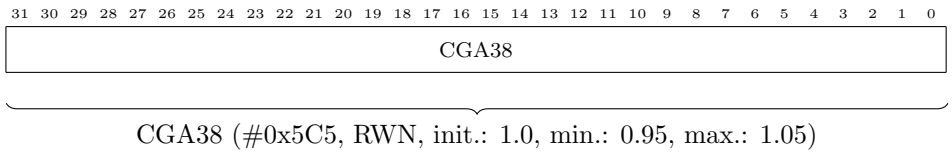
[31:0] CGA36 Calibration gain (Volts/Volts), see DAC calibration.

2.2.197 CGA37 (#0x5C4) - OUT37 calibration gain register



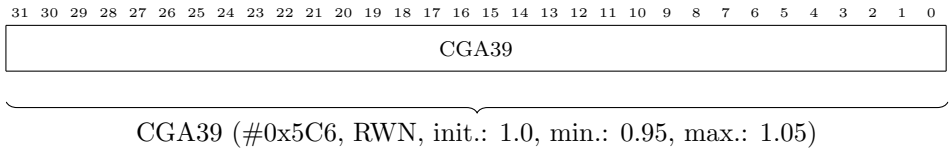
[31:0] CGA37 Calibration gain (Volts/Volts), see DAC calibration.

2.2.198 CGA38 (#0x5C5) - OUT38 calibration gain register



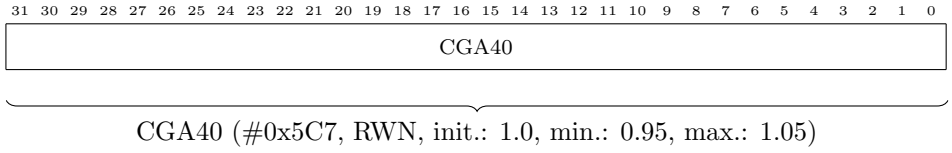
[31:0] CGA38 Calibration gain (Volts/Volts), see DAC calibration.

2.2.199 CGA39 (#0x5C6) - OUT39 calibration gain register



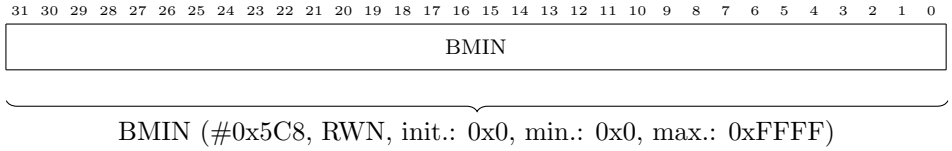
[31:0] CGA39 Calibration gain (Volts/Volts), see DAC calibration.

2.2.200 CGA40 (#0x5C7) - OUT40 calibration gain register



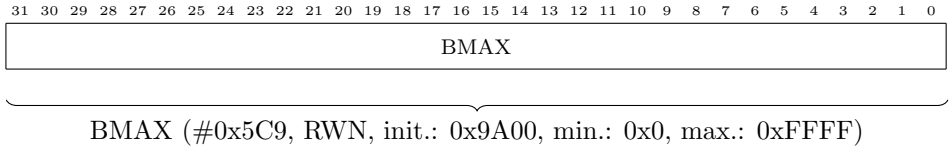
[31:0] CGA40 — Calibration gain (Volts/Volts), see DAC calibration.

2.2.201 BMIN (#0x5C8) - Minimum binary DAC code for all channels register



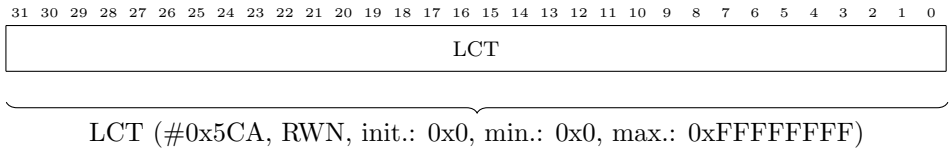
[31:0] BMIN — Minimum DAC value (binary), all channels.

2.2.202 BMAX (#0x5C9) - Maximum binary DAC code for all channels register



[31:0] BMAX — Maximum DAC value (binary), all channels.

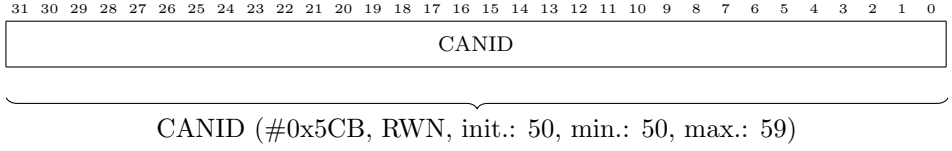
2.2.203 LCT (#0x5CA) - Last calibration time register



[31:0] LCT — Time of the last calibration (UCT) saved as Unix timestamp (compatible to C library time_t).

2.2.204 CANID (#0x5CB) - The node's CAN ID / address register

Writing to this register will change the node's address. The CANopen node will be stopped, initialized and then will automatically enter operational mode using the new address.



[31:0] CANID — The CAN node ID. See Setting the CAN node address for details.

3 DAC calibration

The calibration procedure is used to gain DAC accuracy better than 1 % over entire voltage range.

The calibration procedure requires presence of a calibration device with suitable accuracy class.

Each DAC channel calibration data consists of two registers: calibration offset voltage register (COF_X) and calibration gain register (CGA_X). These two registers are used when calculating the DAC output value for the particular DAC output using the following formula (X stands for DAC channel number):

$$\begin{cases} VCAL_X = VOUT_X \cdot CGA_X + COF_X \\ DAC_X = 2^{16} \cdot \frac{VCAL_X}{5} \end{cases} \tag{5}$$

where:

- $VOUT_X$: the output voltage value commanded by the user
- $VCAL_X$: commanded voltage after calibration (directly fed into DAC)
- DAC_X: the actual value written to DAC
- CGA_X: calibration gain value
- COF_X: calibration offset voltage
- 2^{16}: the DAC resolution

The calibration procedure is used to determine COF_X and CGA_X values for each DAC channel. It is based on a typical two point procedure:

1. The DAC outputs $VCAL_X = 0.5$ V, the actual voltage is measured by the calibration device and stored as V_{X05}.
2. The DAC outputs $VCAL_X = 4.5$ V, the actual voltage is measured by the calibration device and stored as V_{X45}.

The captured values are then used to determine OE_X (offset error) and GE_X (gain error) as follows:

$$\begin{cases} V_{X05} = GE_X \cdot 0.5 + OE_X \\ V_{X45} = GE_X \cdot 4.5 + OE_X \end{cases} \tag{6}$$

therefore:

$$\begin{cases} OE_X = \frac{4.5 \cdot V_{X05} - 0.5 \cdot V_{X45}}{4.5 - 0.5} \\ GE_X = \frac{V_{X05} - OE_X}{0.5} \end{cases} \tag{7}$$

Knowing the offset and gain error for a channel, we can determine COF_X and CGA_X:

$$\begin{cases} V_X = GE_X \cdot VCAL_X + OE_X \\ V_X = GE_X \cdot CGA_X \cdot VOUT_X + GE_X \cdot COF_X + OE_X \\ V_X \equiv VOUT_X \Rightarrow GE_X \cdot CGA_X \equiv 1 \text{ and } GE_X \cdot COF_X + OE_X \equiv 0 \\ CGA_X \equiv \frac{1}{GE_X} \\ COF_X \equiv -\frac{OE_X}{GE_X} \end{cases} \tag{8}$$

This procedure is repeated for all DAC channels and the calculated COF_X and CGA_X register values are written with the calculated values. The registers are then stored in non-volatile memory and used every time the DAC code is calculated.

V. MATES-DIOX-MK1 - variable voltage digital inputs / outputs module

This module is used to interface various types of logic, from 5 V up to 30 V. The device has 20 inputs with switching hysteresis and 20 outputs which provide excellent output power capability reaching 500 mA per pin.

1 Properties

1.1 Functional description

1.1.1 V_X voltage source

The voltage for the inputs / outputs power supply can be selected from three different sources:

- external voltage connected at the rear `5-30 VDC` connector
- internal 24 VDC voltage drawn from the system PSU
- internal 5 VDC voltage from the on-board DC / DC converter

The voltage selection is done by writing the VXS field of the ODEF register.

1.1.2 V_X overload protection

The V_X and GND_X rails are protected by two separate fuses. The fuses rating should be adjusted when the application doesn't use the full power capacity of the device. The factory installed fuses are ultra fast, 10 A devices (standard glass 5x20 mm tubes).

The state of the fuses can be checked without opening the case by reading the VX bit from the SR register. When voltage is applied and none of the two fuses is blown, the VX bit reads as 1. This rule applies to all voltage source settings (see 1.1.1 and 2.2.1). When any of the fuses is blown, or the voltage level is below 4 V, the XV bit reads as 0.

1.1.3 Inputs pull-up / pull-down resistors

The inputs can be configured using JP1 jumper in three ways (requires opening of the case):

- all inputs are high impedance (jumper removed)
- all inputs have pull-up resistors (jumper in 1-2 position)
- all inputs have pull-down resistors (jumper in 2-3 position)

The factory setting is pull-downs enabled.

1.1.4 Outputs mode of operation

Each output is created from a pair of complementary bipolar transistors. The transistors are controlled independently therefore each output can operate in the following modes:

- push-pull operation
- open collector operation
- open emitter operation
- disabled / high impedance

The default setting is push-pull. For open collector and open emitter operation, you have to use external pull-ups or pull-downs with resistance less than ∼10 kΩ. The output mode of operation is set up using OCTRLX registers (see e.g. OCTRL01 (#0x401) - OUT01 control register).

1.2 Electrical parameters

Table 8: Device parameters

Parameter	Symbol	Min.	Typ.	Max.
Current consumption	I_N	150 mA[a]	-	400 mA[b]

[a] at V_{IN} = 24 V, VXS = 11_b, no load
[b] at V_{IN} = 24 V, VXS = 10_b, no load

Table 9: V_X voltage operational parameters

Parameter		Symbol	Min.	Typ.	Max.
Operating voltage		V_X	5 V	-	30 V
V_X idle current consumption[a]	at 5 V	I_{Q5}	-	50 mA	-
	at 12 V	I_{Q12}	-	130 mA	-
	at 24 V	I_{Q24}	-	260 mA	-
	at 30 V	I_{Q30}	-	320 mA	-
Outputs current total[b]	V_X = external 5 V	I_{E5}	-	-	4 A
	V_X = external 12 V	I_{E12}	-	-	4 A
	V_X = external 24 V	I_{E24}	-	-	10 A
	V_X = external 30 V	I_{E30}	-	-	10 A
	V_X = internal 5 V	I_{I5}	-	-	800 mA
	V_X = internal 24 V	I_{I24}	-	-	3 A

[a] All outputs enabled, all inputs in low state
[b] All outputs in high state

Table 10: Digital inputs IN01 - IN20

Parameter	Symbol	Min.	Typ.	Max.
Input voltage	V_I	-0.5 V	-	V_X + 0.5 V
Low level voltage	V_{IL}	-	-	$\frac{1}{3}$ V_X
High level voltage	V_{IH}	$\frac{2}{3}$ V_X	-	-
Pull-up / pull-down	R_{PD}	-	100 kΩ	-

Table 11: Digital outputs OUT01 - OUT20 (V_X = 5 V)

Parameter	Symbol	Min.	Typ.	Max.
Low level voltage[a]	V_L	-	-	0.5 V
High level voltage[b]	V_H	4.5 V	-	-
Output current	I_O	-	-	±120 mA
Transition time	T_t	-	-	20 μs

[a] At I_{IO} = -120 mA
[b] At I_{IO} = +120 mA

Table 12: Digital outputs OUT01 - OUT20 (V_X = 12 V)

Parameter	Symbol	Min.	Typ.	Max.
Low level voltage[a]	V_L	-	-	0.5 V
High level voltage[b]	V_H	11.5 V	-	-
Output current	I_O	-	-	±250 mA
Transition time	T_t	-	-	12 μs

[a] At I_{IO} = -250 mA
[b] At I_{IO} = +250 mA

Table 13: Digital outputs OUT01 - OUT20 (at V_X = 24 V)

Parameter	Symbol	Min.	Typ.	Max.
Low level voltage[a]	V_L	-	-	0.7 V
High level voltage[b]	V_H	23.3 V	-	-
Output current	I_O	-	-	±450 mA
Transition time	T_t	-	-	8 μs

[a] At I_{IO} = -450 mA
[b] At I_{IO} = +450 mA

Table 14: Digital outputs OUT01 - OUT20 (at V_X = 30 V)

Parameter	Symbol	Min.	Typ.	Max.
Low level voltage[a]	V_L	-	-	0.7 V
High level voltage[b]	V_H	29.3 V	-	-
Output current	I_O	-	-	±500 mA
Transition time	T_t	-	-	5 μs

[a] At I_{IO} = -500 mA
[b] At I_{IO} = +500 mA

2 Programming

2.1 Datagrams

2.1.1 Standard status - HEARTBEAT

7	6	5	4	3	2	1	0
STATUS							
HEARTBEAT (NMT = 700_h+ node number)							

[7:0] STATUS Communication status of node, according to CANopen.

2.1.2 Digital inputs state - DIOX-IN

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
I08	I07	I06	I05	I04	I03	I02	I01	I16	I15	I14	I13	I12	I11	I10	I09	RSVD				I20	I19	I18	I17
byte 0								byte 1								byte 2							
DIOX-IN (PDO 1 TX = 180_h+ node number)																							

[0:15] IXX — The current state of the digital inputs (0 represents low electric state)

[16:19] RSVD — Reserved for future use

[20:23] IXX — The current state of the digital inputs (0 represents low electric state)

2.1.3 Digital outputs state - DIOX-OUT

This datagram is used to set the value of the digital outputs of the device. The datagram value can be read back using remote frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
O08	O07	O06	O05	O04	O03	O02	O01	O16	O15	O14	O13	O12	O11	O10	O09	RSVD		MODE		O20	O19	O18	O19
byte 0								byte 1								byte 2							
DIOX-OUT (PDO 1 RX = 200_h+ node number)																							

[0:15] OXX — The current state of the digital outputs (0 represents low electric state).

[16:17] RSVD — Reserved for future use (must be set to zero).

[18:19] MODE — The outputs setting mode:

MODE	Name	Action
00_b	`SET`	OUT ← O01:O20
01_b	`AND`	OUT ← OUT & O01:O20
10_b	`OR`	OUT ← OUT | O01:O20
11_b	`XOR`	OUT ← OUT ^ O01:O20

! → When reading this datagram, the MODE field is always 0.

[20:23] OXX — The current state of the digital outputs (0 represents low electric state).

2.1.4 Individual digital output state - DIOX-XOUT

This datagram is used to set the value of an individual digital output. The datagram can be read back using remote frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CHANNEL								R1	R2	R3	R4	R5	R6	R7	OUT
byte 0								byte 1							
DIOX-OUTX (PDO 2 RX = 300_h+ node number)															

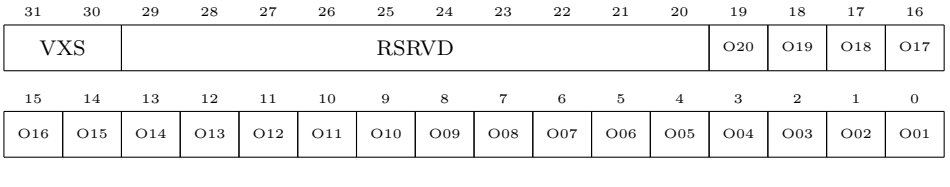
Bits 0 to 7 — The channel number to set (0 means OUT01, 19 means OUT20).

Bits 8 to 14 — Reserved for future use (set to 0).

Bit 15 — Output value.

2.2 Registers

2.2.1 ODEF (#0x400) - Outputs default value register.



ODEF (#0x400, RWN, init.: 0xC0000000, min.: 0x0, max.: 0xC00FFFFF)

[31:30] VXS — V_X voltage selection:

- 00_b - V_X comes from the external voltage connector 5-30 VDC
- 10_b - V_X comes from the system PSU (24 V)
- 11_b (default) - V_X comes from the internal 5 V voltage regulator
- 01_b - invalid

→ The 11_b setting cannot be applied (changed to) when external voltage is present at the 5-30 VDC connector. This is done to protect internal electronics from a voltage spike during switching. To be able to set VXS to this value, turn off or disconnect the external voltage.

→ Changing VXS may take over two seconds.

[29:20] RSRVD — Reserved, do not use.

[19] O20 — `OUT20` default value after startup.

[18] O19 — `OUT19` default value after startup.

[17] O18 — `OUT18` default value after startup.

[16] O17 — `OUT17` default value after startup.

[15] O16 — `OUT16` default value after startup.

[14] O15 — `OUT15` default value after startup.

[13] O14 — `OUT14` default value after startup.

[12] O13 — `OUT13` default value after startup.

[11] O12 — `OUT12` default value after startup.

[10] O11 — `OUT11` default value after startup.

[9] O10 — `OUT10` default value after startup.

[8] O09 — `OUT09` default value after startup.

[7] O08 — `OUT08` default value after startup.

[6] O07 — `OUT07` default value after startup.

[5] O06 — `OUT06` default value after startup.

[4] O05 — `OUT05` default value after startup.

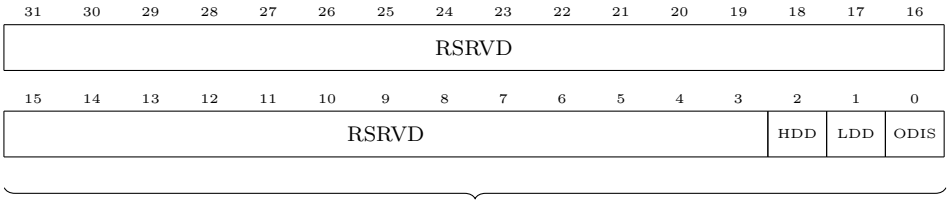
[3] O04 — `OUT04` default value after startup.

[2] O03 — `OUT03` default value after startup.

[1] O02 — `OUT02` default value after startup.

[0] O01 — `OUT01` default value after startup.

2.2.2 OCTRL01 (#0x401) - OUT01 control register



OCTRL01 (#0x401, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

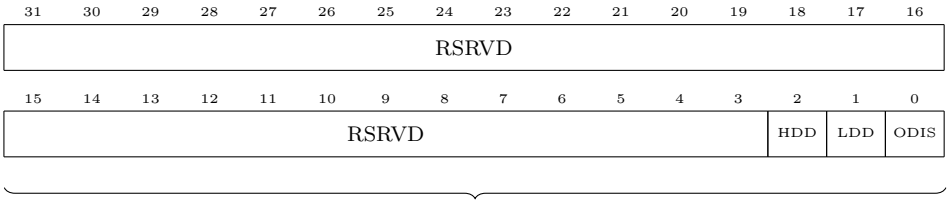
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.3 OCTRL02 (#0x402) - OUT02 control register



OCTRL02 (#0x402, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

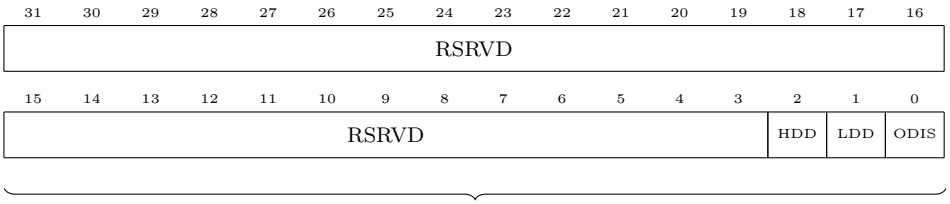
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.4 OCTRL03 (#0x403) - OUT03 control register



OCTRL03 (#0x403, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

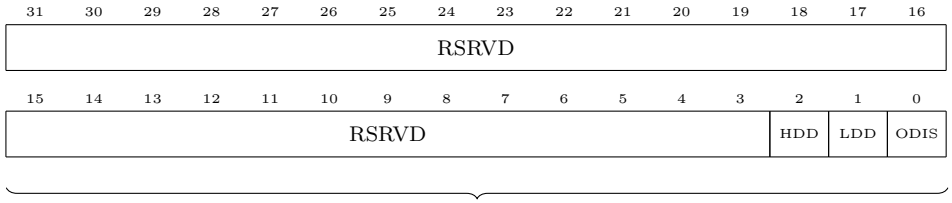
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.5 OCTRL04 (#0x404) - OUT04 control register



OCTRL04 (#0x404, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

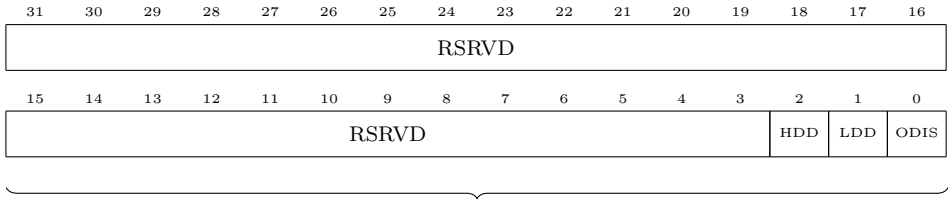
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.6 OCTRL05 (#0x405) - OUT05 control register



OCTRL05 (#0x405, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

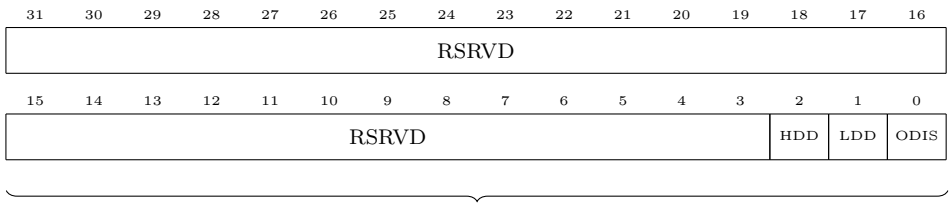
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.7 OCTRL06 (#0x406) - OUT06 control register



OCTRL06 (#0x406, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

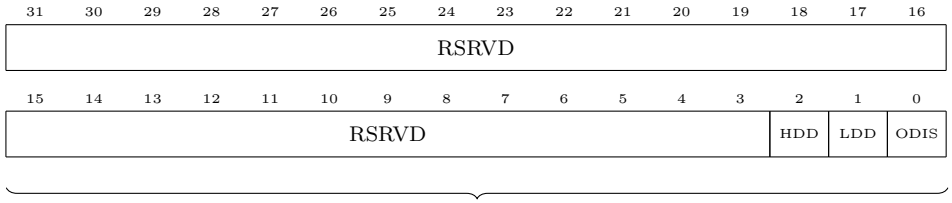
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.8 OCTRL07 (#0x407) - OUT07 control register



OCTRL07 (#0x407, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

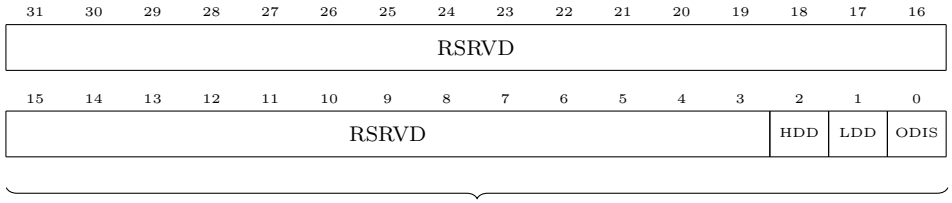
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.9 OCTRL08 (#0x408) - OUT08 control register



OCTRL08 (#0x408, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

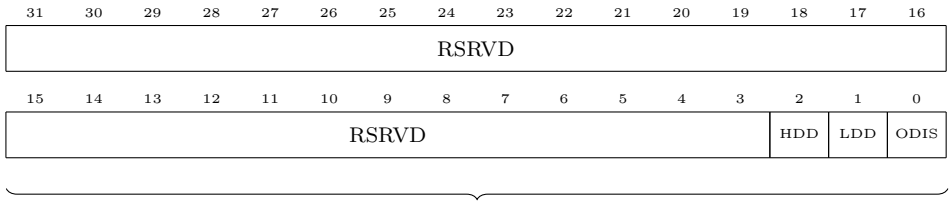
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.10 OCTRL09 (#0x409) - OUT09 control register



OCTRL09 (#0x409, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

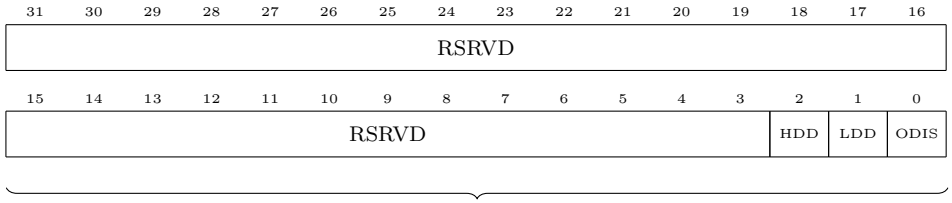
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.11 OCTRL10 (#0x40A) - OUT10 control register



OCTRL10 (#0x40A, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

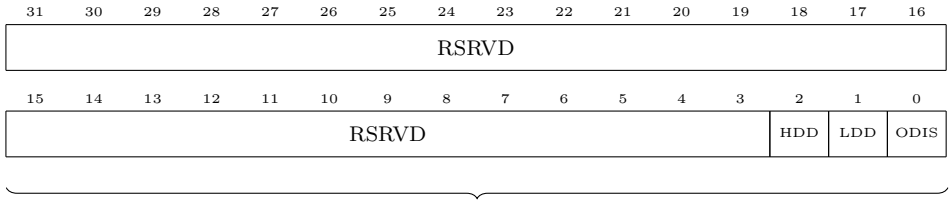
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.12 OCTRL11 (#0x40B) - OUT11 control register



OCTRL11 (#0x40B, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

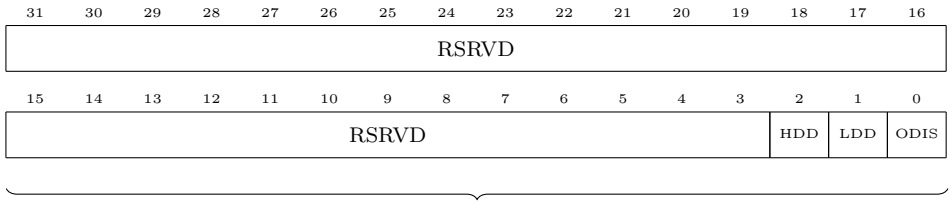
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.13 OCTRL12 (#0x40C) - OUT12 control register



OCTRL12 (#0x40C, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

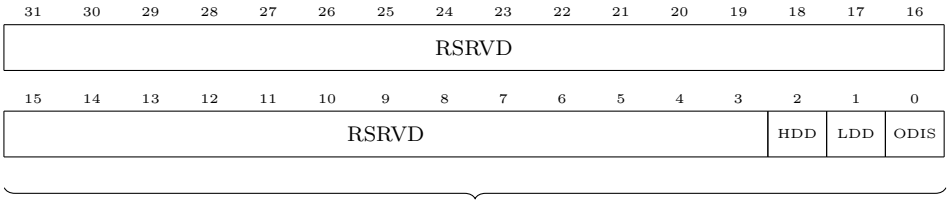
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.14 OCTRL13 (#0x40D) - OUT13 control register



OCTRL13 (#0x40D, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

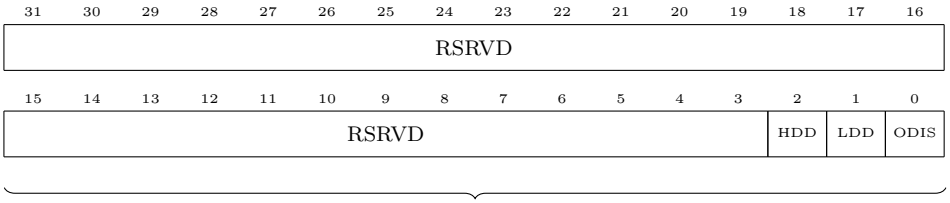
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.15 OCTRL14 (#0x40E) - OUT14 control register



OCTRL14 (#0x40E, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

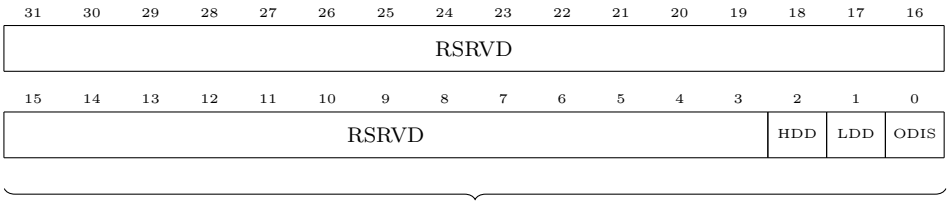
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.16 OCTRL15 (#0x40F) - OUT15 control register



OCTRL15 (#0x40F, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

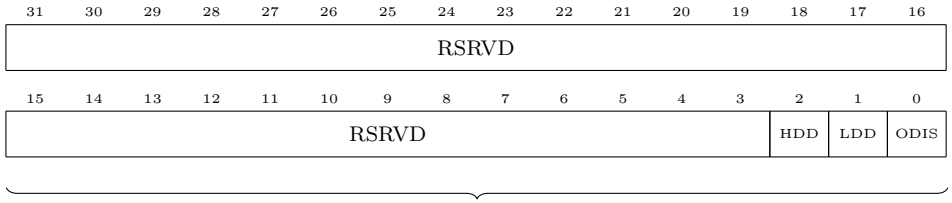
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.17 OCTRL16 (#0x410) - OUT16 control register



OCTRL16 (#0x410, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

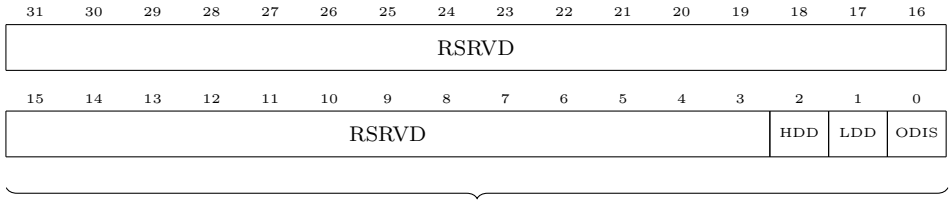
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.18 OCTRL17 (#0x411) - OUT17 control register



OCTRL17 (#0x411, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

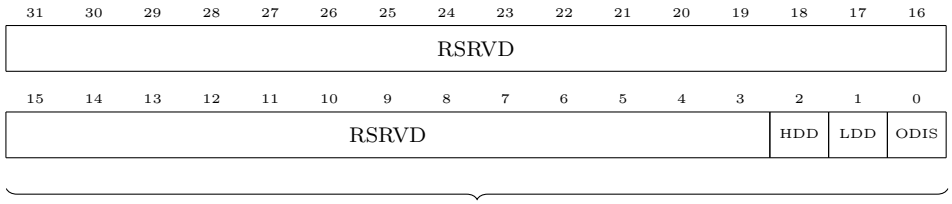
[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

[0] ODIS Output disable (output enters high impedance state).

2.2.19 OCTRL18 (#0x412) - OUT18 control register



OCTRL18 (#0x412, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

[31:3] RSRVD Reserved, do not use.

[2] HDD High driver disable (allows open collector operation).

[1] LDD Low driver disable (allows open emitter operation).

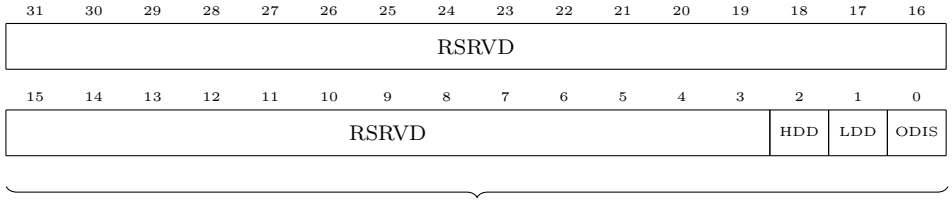
[0] ODIS Output disable (output enters high impedance state).

2.2.20 OCTRL19 (#0x413) - OUT19 control register

OCTRL19 (#0x413, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

[31:3]	RSRVD	Reserved, do not use.
[2]	HDD	High driver disable (allows open collector operation).
[1]	LDD	Low driver disable (allows open emitter operation).
[0]	ODIS	Output disable (output enters high impedance state).

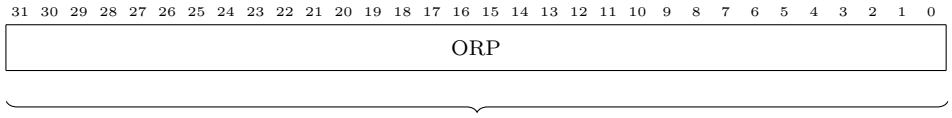
2.2.21 OCTRL20 (#0x414) - OUT20 control register



OCTRL20 (#0x414, RWN, init.: 0x0, min.: 0x0, max.: 0x7)

[31:3]	RSRVD	Reserved, do not use.
[2]	HDD	High driver disable (allows open collector operation).
[1]	LDD	Low driver disable (allows open emitter operation).
[0]	ODIS	Output disable (output enters high impedance state).

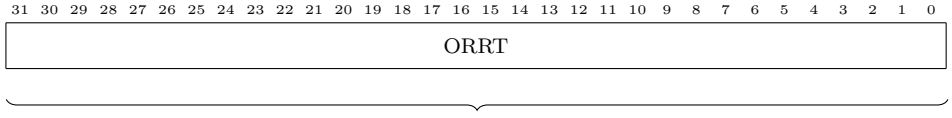
2.2.22 ORP (#0x415) - Outputs restore period register



ORP (#0x415, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0]	ORP	Outputs restore period (microseconds). When set to a non-zero value, defines the time after which all outputs are set to their default values if no output setting command is executed. Writing this register causes ORRT register to be reloaded.

2.2.23 ORRT (#0x416) - Outptus restore remaining time register



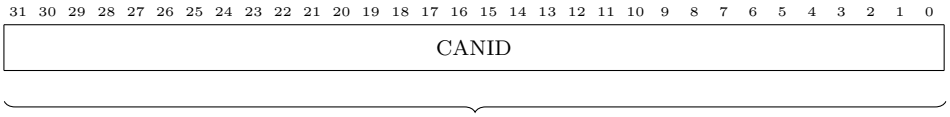
ORRT (#0x416, RO, init.: 0x0)

[31:0]	ORRT	Outputs restore remaining time (microseconds). Represents the time remaining before all outputs are restored to their default value. This register is reloaded with the value of ORP each time the outputs state is being set (and also when ORP is written). When ORP is non-zero and the value of ORRT drops to zero, the following actions are performed:

- all signal generators are disabled by setting their PRDx registers to 0;
- the outputs are set to the value of ODEF register.

2.2.24 CANID (#0x417) - The node's CAN ID / address register

Writing to this register will change the node's address. The CANopen node will be stopped, initialized and then will automatically enter operational mode using the new address.



CANID (#0x417, RWN, init.: 40, min.: 40, max.: 49)

[31:0] CANID — The CAN node ID. See Setting the CAN node address for details.

VI. MATES-ADC5-MK1 - analogue inputs module

This module is used to measure analogue signals. It provides 40 single ended ADC channels with the voltage range of -10 to 10 V.

1 Properties

1.1 Electrical parameters

Table 15: Device parameters

Parameter	Symbol	Min.	Typ.	Max.
Current consumption[a]	I_N	-	-	150 mA

[a] at $V_{IN} = 24$ V

Table 16: Analogue inputs IN01 - IN40

Parameter	Symbol	Min.	Typ.	Max.
Input voltage	V_I	-10 V	-	10 V
Input impedance	Z_I	-	470 kΩ	-
Resolution	N	-	2^{16}	-
Step response latency[a]	T_{dly}	-	-	10 ms
DC accuracy[b]	Δ_V	-	-	0.2 % ±10 mV

[a] V_{IN} step from 0 V to 10 V, takes input filter delay and worst case ADC sequencer delay into account
[b] guaranteed within 6 months after calibration over entire ±10 V range

2 Programming

2.1 Datagrams

2.1.1 Standard status - HEARTBEAT

7	6	5	4	3	2	1	0
STATUS							
HEARTBEAT (NMT = 700_h+ node number)							

[7:0] STATUS — Communication status of node, according to CANopen.

2.1.2 Analogue input value - ADC5-IN

This datagram is used to get the input voltage value of an individual ADC input. This datagram contains raw ADC readouts ignoring the calibration data. You have to set the channel to read from before using this datagram (see ADC channel selection - ADC5-CS).

The theoretical voltage value can be calculated from the binary ADC code as follows:

$$V_{ADC} = -\frac{4.096 \cdot \frac{X}{2^{16}} - 2.051}{0.195} \tag{9}$$

Where:

V_{ADC}: the voltage at ADC input in Volts

X: the binary ADC value read

2^{16}: the ADC resolution

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23
CHANNEL	LSB	MSB
byte 0	byte 1	byte 2
ADC5-IN (PDO 1 TX = 180_h+ node number)		

[0:7] CHANNEL ADC channel indicator (0 means IN01, 39 means IN40).

[8:15] LSB ADC binary value, less significant byte.

[16:23] MSB ADC binary value, more significant byte.

2.1.3 Analogue input voltage - ADC5-VIN

This datagram is used to get the input voltage for an individual ADC input. The provided voltage value is identical to the actual value at the input. The readout takes calibration data into account. You have to set the channel to read from before using this datagram (see ADC channel selection - ADC5-CS).

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32 33 34 35 36 37 38 39
CHANNEL	VALUE (LE)	
byte 0	byte 1	byte 2	byte 3	byte 4
ADC5-VIN (PDO 2 TX = 280_h+ node number)		

[0:7] CHANNEL ADC channel indicator (0 means OUT01, 39 means OUT40).

[8:39] VALUE DAC voltage value [Volts], stored as IEEE-754 single precision floating point number in Little Endian.

2.1.4 ADC channel selection - ADC5-CS

The datagram is used to set the ADC channel number for the subsequent read operations. The datagram can be read back using remote frame.

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15
CHANNEL	RSVD
ADC5-CS (PDO 1 RX = 200_h+ node number)	

[0:7] CHANNEL ADC channel selector (0 means IN01, 39 means IN40).

[8:15] RSVD Reserved for future use.

2.2 Registers

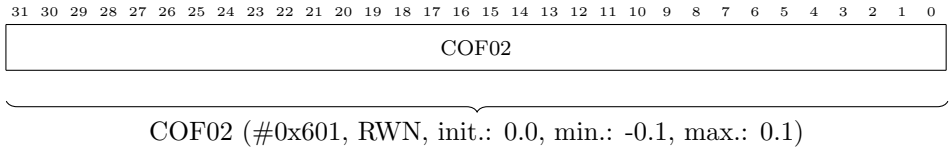
2.2.1 COF01 (#0x600) - IN01 calibration offset register

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
COF01

COF01 (#0x600, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF01 Calibration offset (Volts), see ADC calibration.

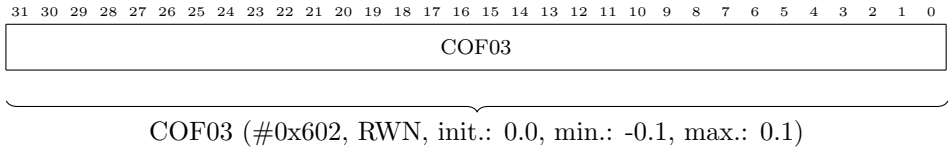
2.2.2 COF02 (#0x601) - IN02 calibration offset register



COF02 (#0x601, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF02 Calibration offset (Volts), see ADC calibration.

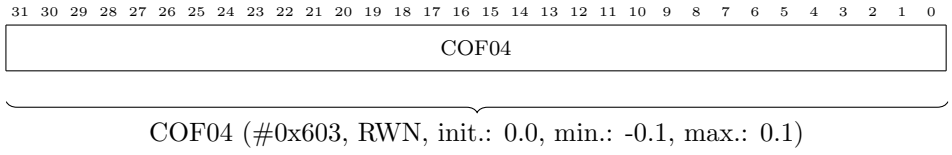
2.2.3 COF03 (#0x602) - IN03 calibration offset register



COF03 (#0x602, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF03 Calibration offset (Volts), see ADC calibration.

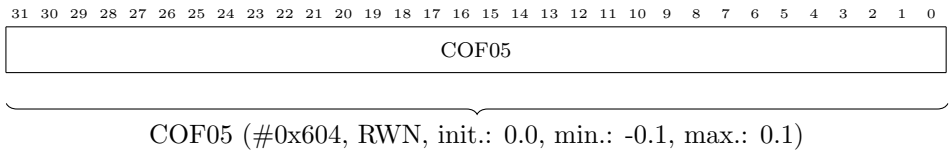
2.2.4 COF04 (#0x603) - IN04 calibration offset register



COF04 (#0x603, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF04 Calibration offset (Volts), see ADC calibration.

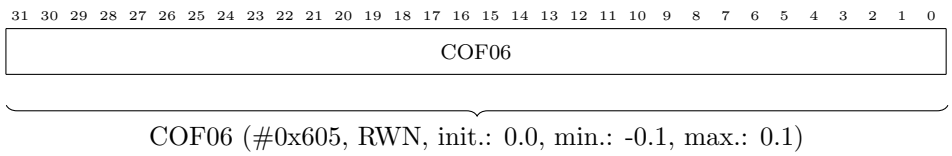
2.2.5 COF05 (#0x604) - IN05 calibration offset register



COF05 (#0x604, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF05 Calibration offset (Volts), see ADC calibration.

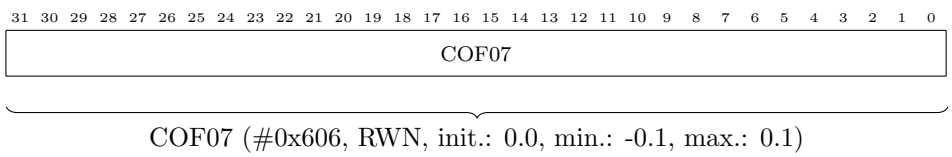
2.2.6 COF06 (#0x605) - IN06 calibration offset register



COF06 (#0x605, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF06 Calibration offset (Volts), see ADC calibration.

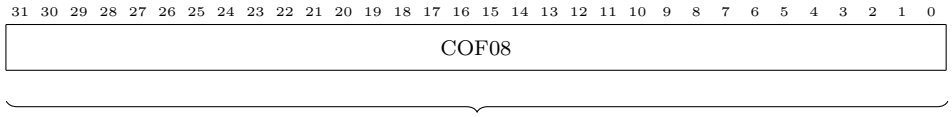
2.2.7 COF07 (#0x606) - IN07 calibration offset register



COF07 (#0x606, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF07 Calibration offset (Volts), see ADC calibration.

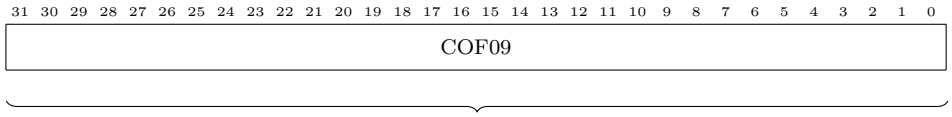
2.2.8 COF08 (#0x607) - IN08 calibration offset register



COF08 (#0x607, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF08 Calibration offset (Volts), see ADC calibration.

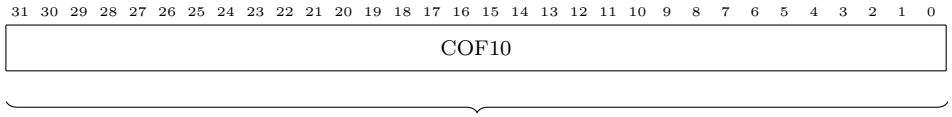
2.2.9 COF09 (#0x608) - IN09 calibration offset register



COF09 (#0x608, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF09 Calibration offset (Volts), see ADC calibration.

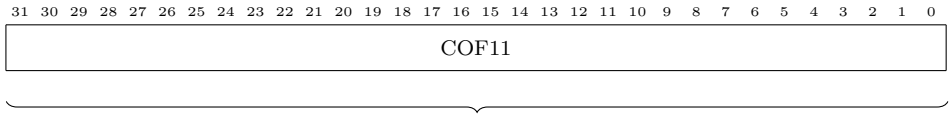
2.2.10 COF10 (#0x609) - IN10 calibration offset register



COF10 (#0x609, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF10 Calibration offset (Volts), see ADC calibration.

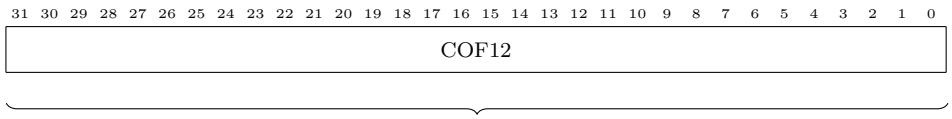
2.2.11 COF11 (#0x60A) - IN11 calibration offset register



COF11 (#0x60A, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF11 Calibration offset (Volts), see ADC calibration.

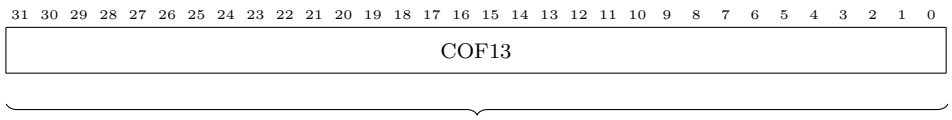
2.2.12 COF12 (#0x60B) - IN12 calibration offset register



COF12 (#0x60B, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF12 Calibration offset (Volts), see ADC calibration.

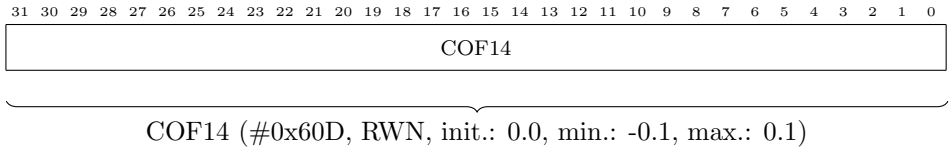
2.2.13 COF13 (#0x60C) - IN13 calibration offset register



COF13 (#0x60C, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF13 Calibration offset (Volts), see ADC calibration.

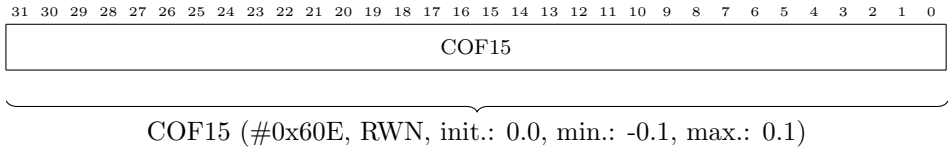
2.2.14 COF14 (#0x60D) - IN14 calibration offset register



COF14 (#0x60D, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF14 Calibration offset (Volts), see ADC calibration.

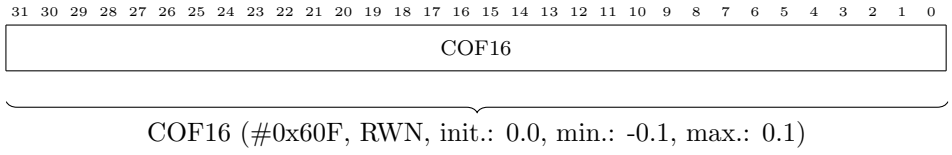
2.2.15 COF15 (#0x60E) - IN15 calibration offset register



COF15 (#0x60E, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF15 Calibration offset (Volts), see ADC calibration.

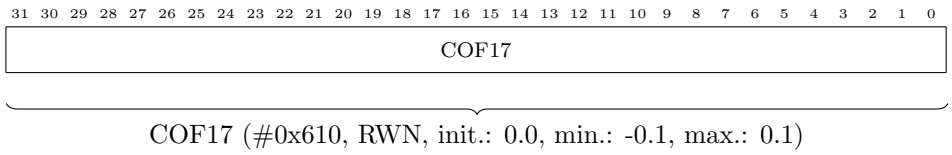
2.2.16 COF16 (#0x60F) - IN16 calibration offset register



COF16 (#0x60F, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF16 Calibration offset (Volts), see ADC calibration.

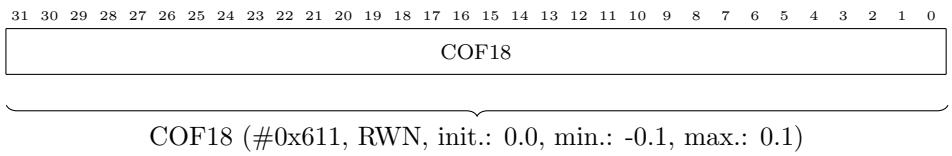
2.2.17 COF17 (#0x610) - IN17 calibration offset register



COF17 (#0x610, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF17 Calibration offset (Volts), see ADC calibration.

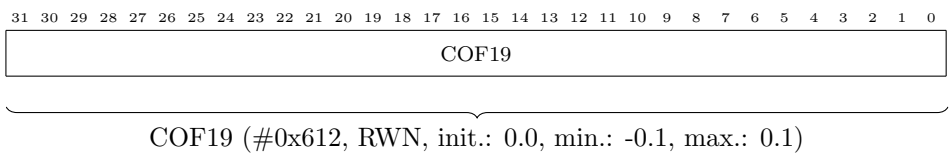
2.2.18 COF18 (#0x611) - IN18 calibration offset register



COF18 (#0x611, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF18 Calibration offset (Volts), see ADC calibration.

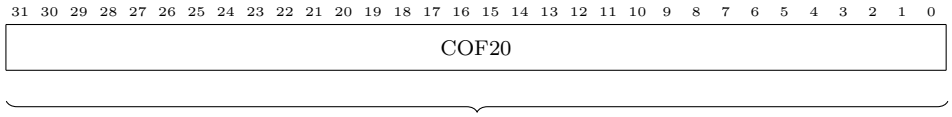
2.2.19 COF19 (#0x612) - IN19 calibration offset register



COF19 (#0x612, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF19 Calibration offset (Volts), see ADC calibration.

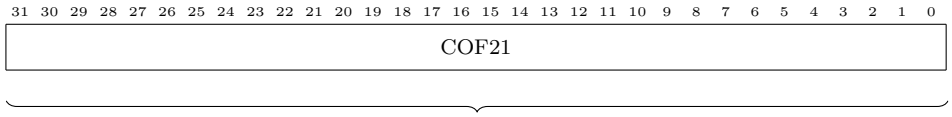
2.2.20 COF20 (#0x613) - IN20 calibration offset register



COF20 (#0x613, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF20 Calibration offset (Volts), see ADC calibration.

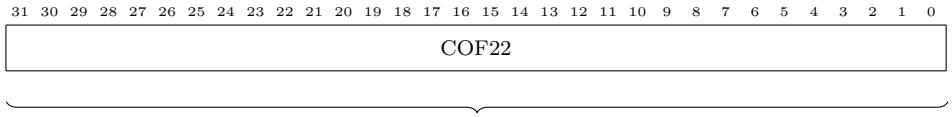
2.2.21 COF21 (#0x614) - IN21 calibration offset register



COF21 (#0x614, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF21 Calibration offset (Volts), see ADC calibration.

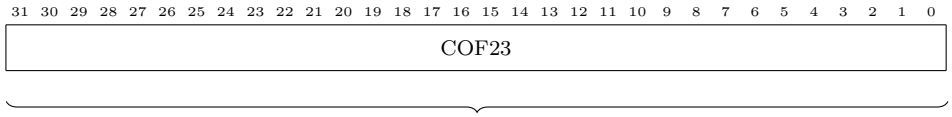
2.2.22 COF22 (#0x615) - IN22 calibration offset register



COF22 (#0x615, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF22 Calibration offset (Volts), see ADC calibration.

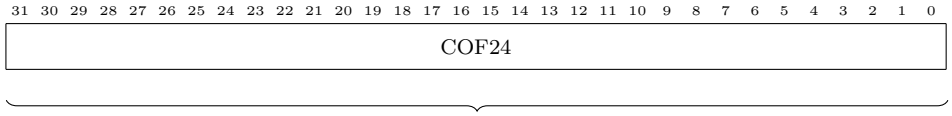
2.2.23 COF23 (#0x616) - IN23 calibration offset register



COF23 (#0x616, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF23 Calibration offset (Volts), see ADC calibration.

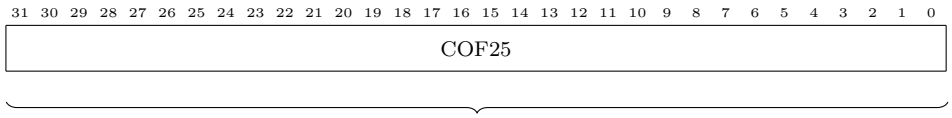
2.2.24 COF24 (#0x617) - IN24 calibration offset register



COF24 (#0x617, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF24 Calibration offset (Volts), see ADC calibration.

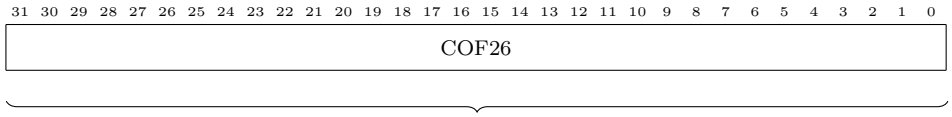
2.2.25 COF25 (#0x618) - IN25 calibration offset register



COF25 (#0x618, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF25 Calibration offset (Volts), see ADC calibration.

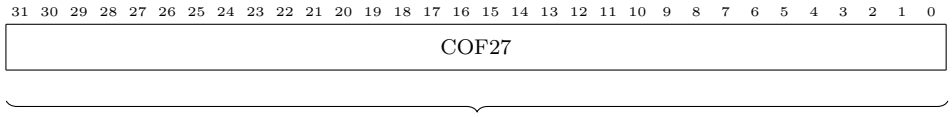
2.2.26 COF26 (#0x619) - IN26 calibration offset register



COF26 (#0x619, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF26 Calibration offset (Volts), see ADC calibration.

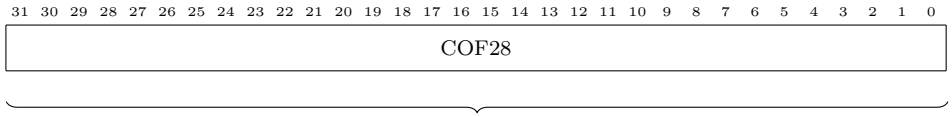
2.2.27 COF27 (#0x61A) - IN27 calibration offset register



COF27 (#0x61A, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF27 Calibration offset (Volts), see ADC calibration.

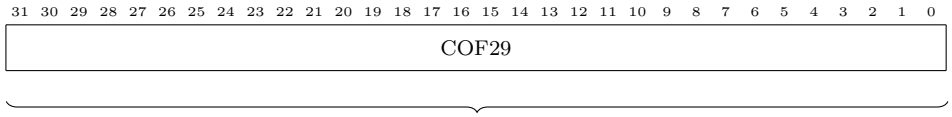
2.2.28 COF28 (#0x61B) - IN28 calibration offset register



COF28 (#0x61B, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF28 Calibration offset (Volts), see ADC calibration.

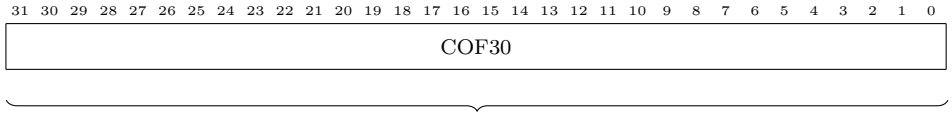
2.2.29 COF29 (#0x61C) - IN29 calibration offset register



COF29 (#0x61C, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF29 Calibration offset (Volts), see ADC calibration.

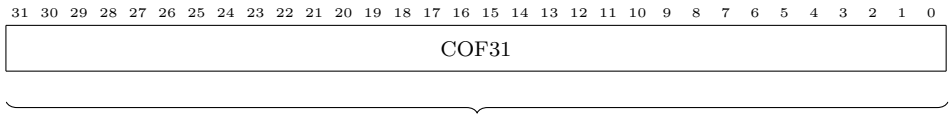
2.2.30 COF30 (#0x61D) - IN30 calibration offset register



COF30 (#0x61D, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF30 Calibration offset (Volts), see ADC calibration.

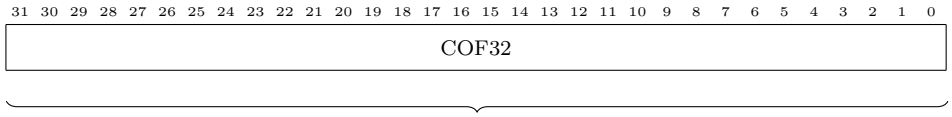
2.2.31 COF31 (#0x61E) - IN31 calibration offset register



COF31 (#0x61E, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF31 Calibration offset (Volts), see ADC calibration.

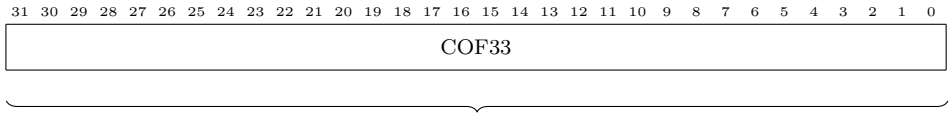
2.2.32 COF32 (#0x61F) - IN32 calibration offset register



COF32 (#0x61F, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF32 Calibration offset (Volts), see ADC calibration.

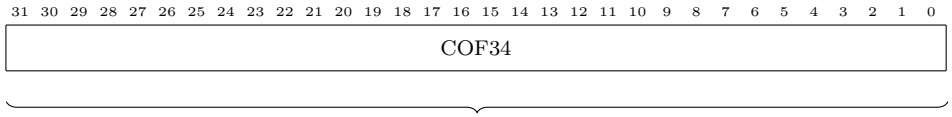
2.2.33 COF33 (#0x620) - IN33 calibration offset register



COF33 (#0x620, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF33 Calibration offset (Volts), see ADC calibration.

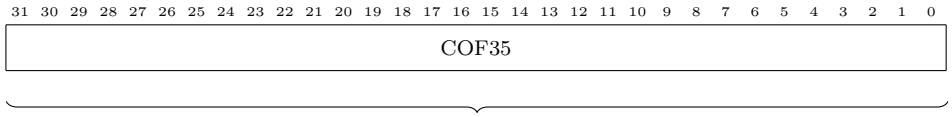
2.2.34 COF34 (#0x621) - IN34 calibration offset register



COF34 (#0x621, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF34 Calibration offset (Volts), see ADC calibration.

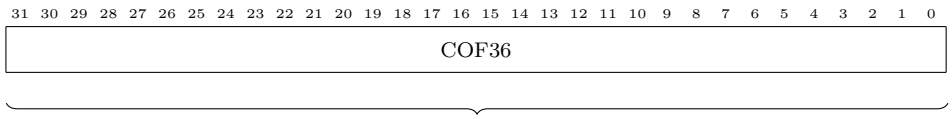
2.2.35 COF35 (#0x622) - IN35 calibration offset register



COF35 (#0x622, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF35 Calibration offset (Volts), see ADC calibration.

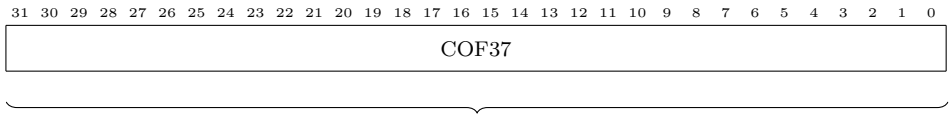
2.2.36 COF36 (#0x623) - IN36 calibration offset register



COF36 (#0x623, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF36 Calibration offset (Volts), see ADC calibration.

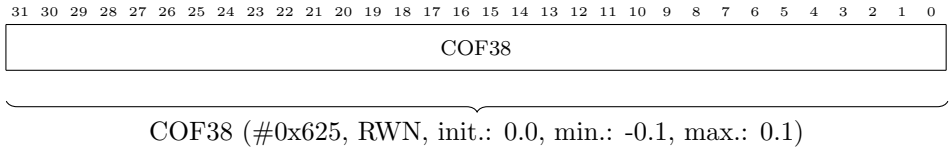
2.2.37 COF37 (#0x624) - IN37 calibration offset register



COF37 (#0x624, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF37 Calibration offset (Volts), see ADC calibration.

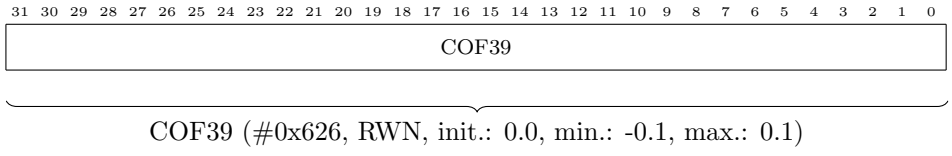
2.2.38 COF38 (#0x625) - IN38 calibration offset register



COF38 (#0x625, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF38 Calibration offset (Volts), see ADC calibration.

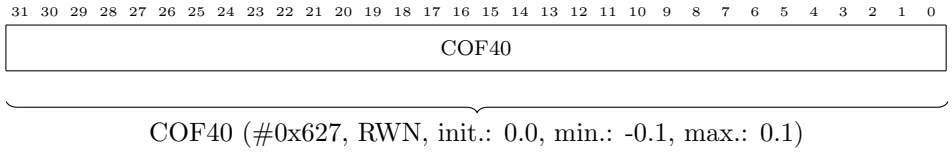
2.2.39 COF39 (#0x626) - IN39 calibration offset register



COF39 (#0x626, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF39 Calibration offset (Volts), see ADC calibration.

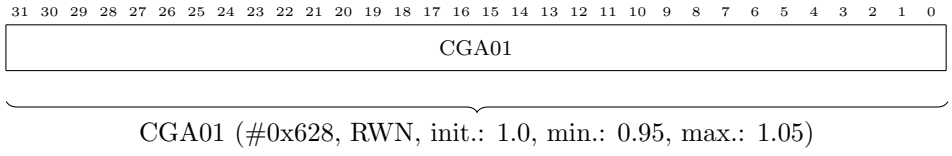
2.2.40 COF40 (#0x627) - IN40 calibration offset register



COF40 (#0x627, RWN, init.: 0.0, min.: -0.1, max.: 0.1)

[31:0] COF40 Calibration offset (Volts), see ADC calibration.

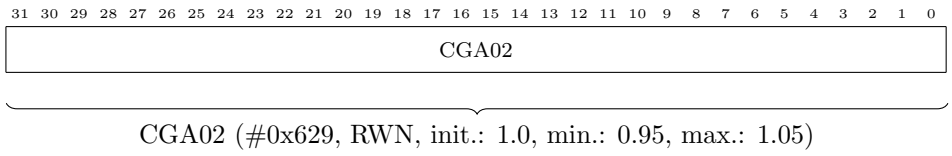
2.2.41 CGA01 (#0x628) - IN01 calibration gain register



CGA01 (#0x628, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA01 Calibration gain (Volts/Volts), see ADC calibration.

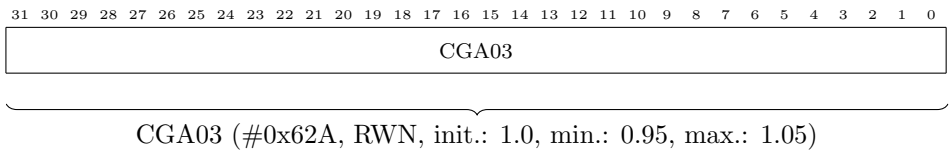
2.2.42 CGA02 (#0x629) - IN02 calibration gain register



CGA02 (#0x629, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA02 Calibration gain (Volts/Volts), see ADC calibration.

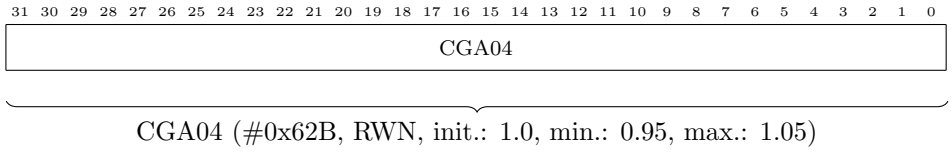
2.2.43 CGA03 (#0x62A) - IN03 calibration gain register



CGA03 (#0x62A, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA03 Calibration gain (Volts/Volts), see ADC calibration.

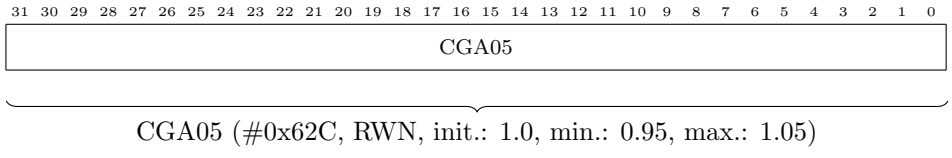
2.2.44 CGA04 (#0x62B) - IN04 calibration gain register



CGA04 (#0x62B, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA04 Calibration gain (Volts/Volts), see ADC calibration.

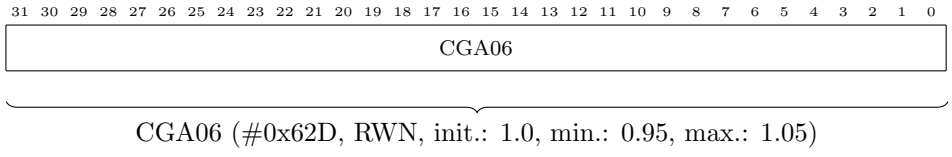
2.2.45 CGA05 (#0x62C) - IN05 calibration gain register



CGA05 (#0x62C, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA05 Calibration gain (Volts/Volts), see ADC calibration.

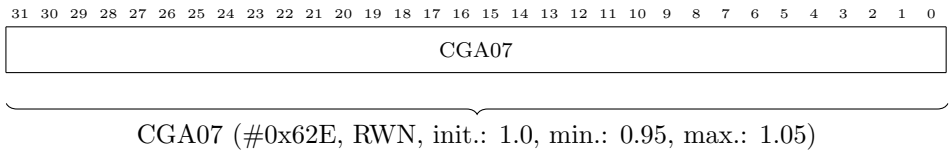
2.2.46 CGA06 (#0x62D) - IN06 calibration gain register



CGA06 (#0x62D, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA06 Calibration gain (Volts/Volts), see ADC calibration.

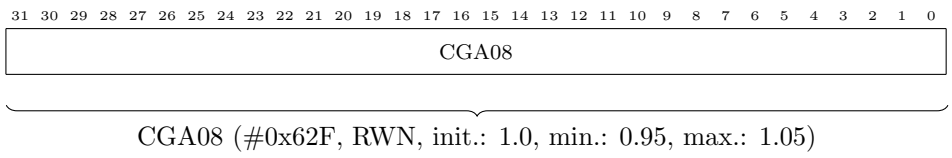
2.2.47 CGA07 (#0x62E) - IN07 calibration gain register



CGA07 (#0x62E, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA07 Calibration gain (Volts/Volts), see ADC calibration.

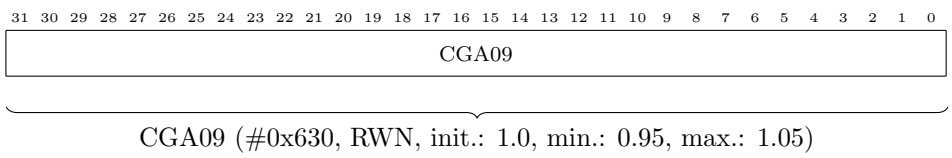
2.2.48 CGA08 (#0x62F) - IN08 calibration gain register



CGA08 (#0x62F, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA08 Calibration gain (Volts/Volts), see ADC calibration.

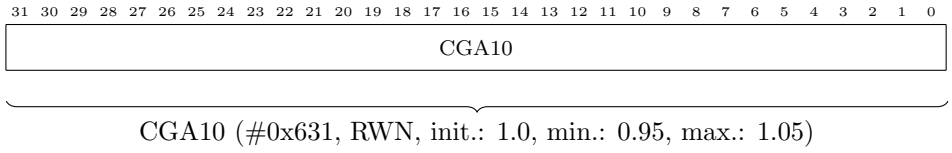
2.2.49 CGA09 (#0x630) - IN09 calibration gain register



CGA09 (#0x630, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA09 Calibration gain (Volts/Volts), see ADC calibration.

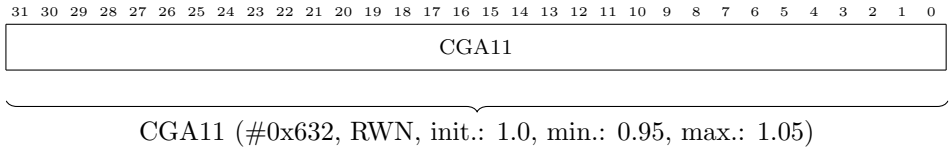
2.2.50 CGA10 (#0x631) - IN10 calibration gain register



CGA10 (#0x631, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA10 Calibration gain (Volts/Volts), see ADC calibration.

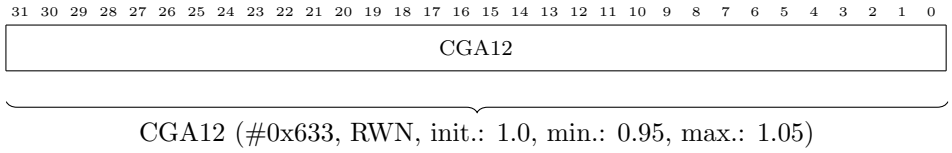
2.2.51 CGA11 (#0x632) - IN11 calibration gain register



CGA11 (#0x632, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA11 Calibration gain (Volts/Volts), see ADC calibration.

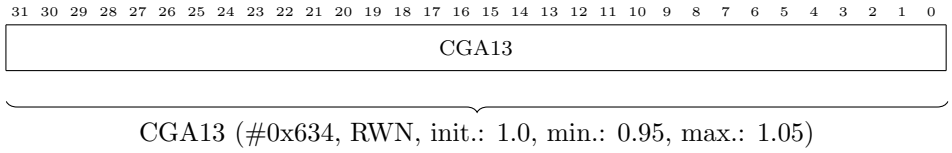
2.2.52 CGA12 (#0x633) - IN12 calibration gain register



CGA12 (#0x633, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA12 Calibration gain (Volts/Volts), see ADC calibration.

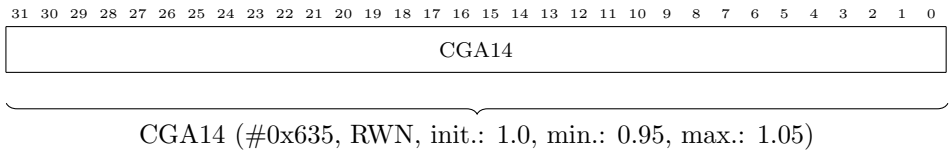
2.2.53 CGA13 (#0x634) - IN13 calibration gain register



CGA13 (#0x634, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA13 Calibration gain (Volts/Volts), see ADC calibration.

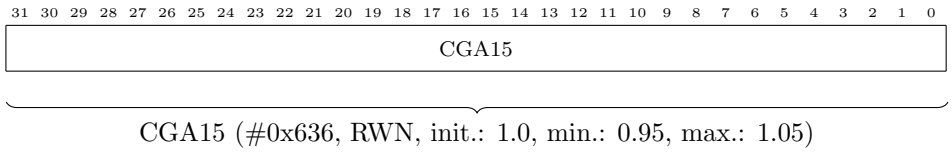
2.2.54 CGA14 (#0x635) - IN14 calibration gain register



CGA14 (#0x635, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA14 Calibration gain (Volts/Volts), see ADC calibration.

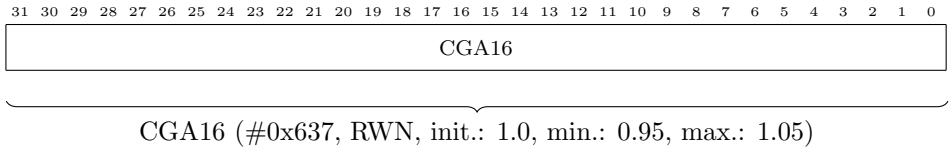
2.2.55 CGA15 (#0x636) - IN15 calibration gain register



CGA15 (#0x636, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA15 Calibration gain (Volts/Volts), see ADC calibration.

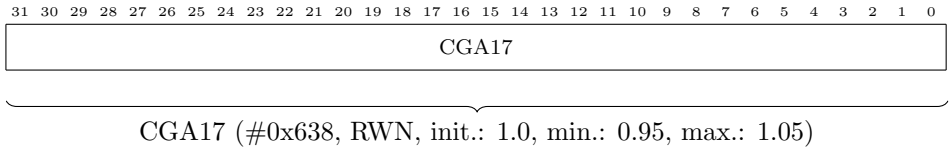
2.2.56 CGA16 (#0x637) - IN16 calibration gain register



CGA16 (#0x637, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA16 Calibration gain (Volts/Volts), see ADC calibration.

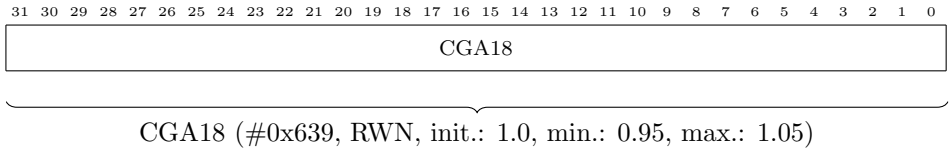
2.2.57 CGA17 (#0x638) - IN17 calibration gain register



CGA17 (#0x638, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA17 Calibration gain (Volts/Volts), see ADC calibration.

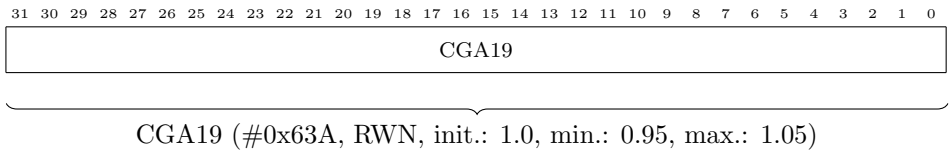
2.2.58 CGA18 (#0x639) - IN18 calibration gain register



CGA18 (#0x639, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA18 Calibration gain (Volts/Volts), see ADC calibration.

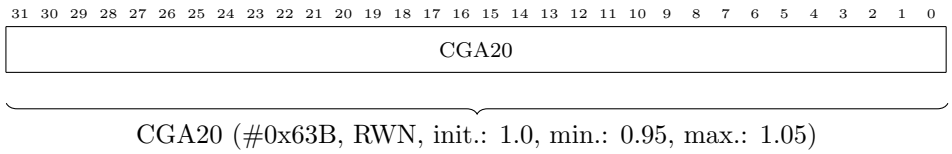
2.2.59 CGA19 (#0x63A) - IN19 calibration gain register



CGA19 (#0x63A, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA19 Calibration gain (Volts/Volts), see ADC calibration.

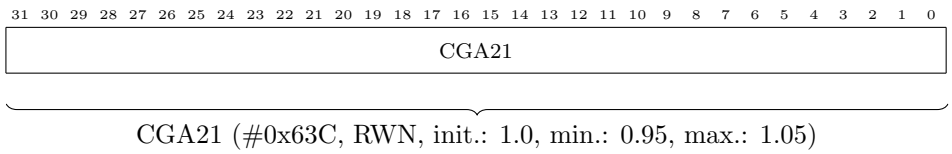
2.2.60 CGA20 (#0x63B) - IN20 calibration gain register



CGA20 (#0x63B, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA20 Calibration gain (Volts/Volts), see ADC calibration.

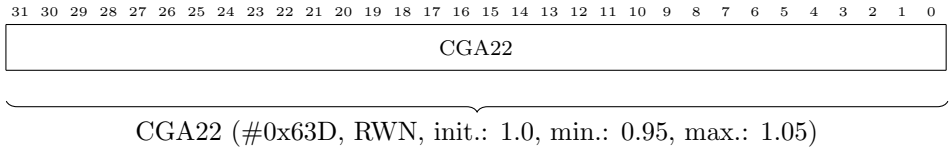
2.2.61 CGA21 (#0x63C) - IN21 calibration gain register



CGA21 (#0x63C, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

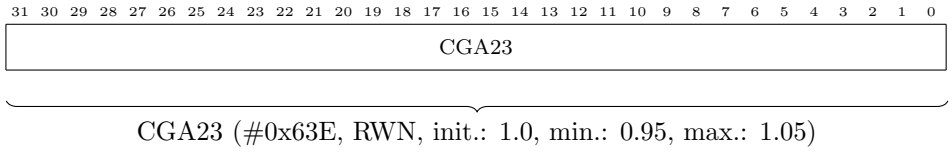
[31:0] CGA21 Calibration gain (Volts/Volts), see ADC calibration.

2.2.62 CGA22 (#0x63D) - IN22 calibration gain register



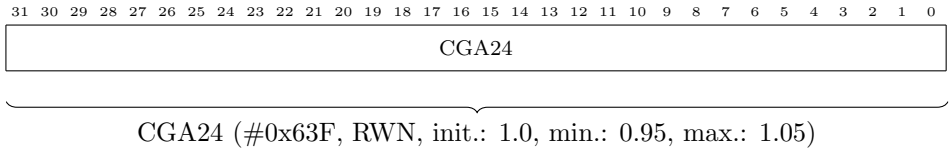
[31:0] CGA22 Calibration gain (Volts/Volts), see ADC calibration.

2.2.63 CGA23 (#0x63E) - IN23 calibration gain register



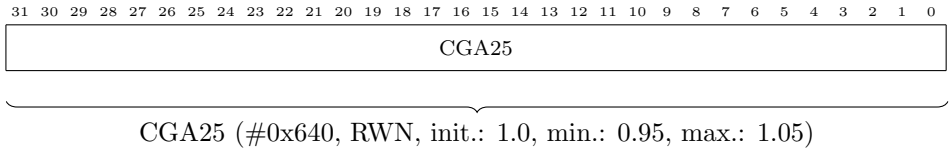
[31:0] CGA23 Calibration gain (Volts/Volts), see ADC calibration.

2.2.64 CGA24 (#0x63F) - IN24 calibration gain register



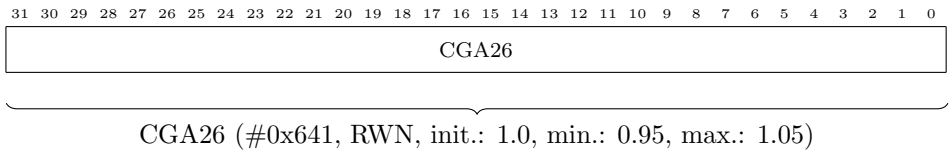
[31:0] CGA24 Calibration gain (Volts/Volts), see ADC calibration.

2.2.65 CGA25 (#0x640) - IN25 calibration gain register



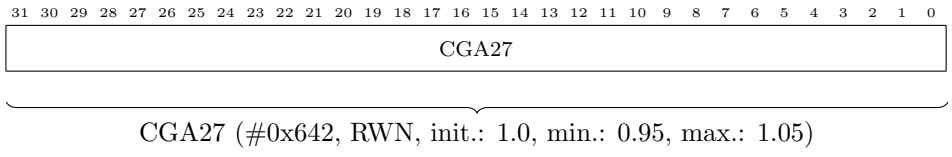
[31:0] CGA25 Calibration gain (Volts/Volts), see ADC calibration.

2.2.66 CGA26 (#0x641) - IN26 calibration gain register



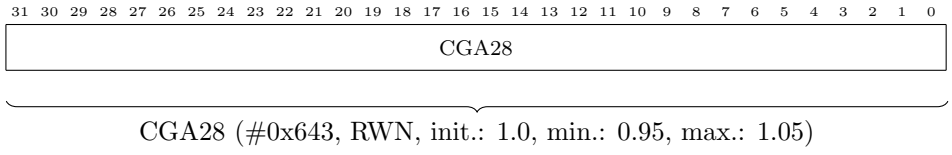
[31:0] CGA26 Calibration gain (Volts/Volts), see ADC calibration.

2.2.67 CGA27 (#0x642) - IN27 calibration gain register



[31:0] CGA27 Calibration gain (Volts/Volts), see ADC calibration.

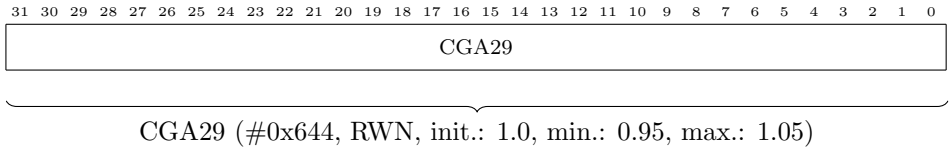
2.2.68 CGA28 (#0x643) - IN28 calibration gain register



CGA28 (#0x643, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA28 Calibration gain (Volts/Volts), see ADC calibration.

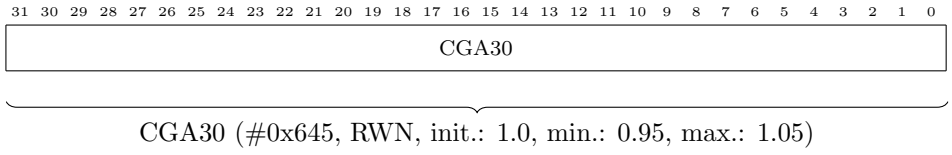
2.2.69 CGA29 (#0x644) - IN29 calibration gain register



CGA29 (#0x644, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA29 Calibration gain (Volts/Volts), see ADC calibration.

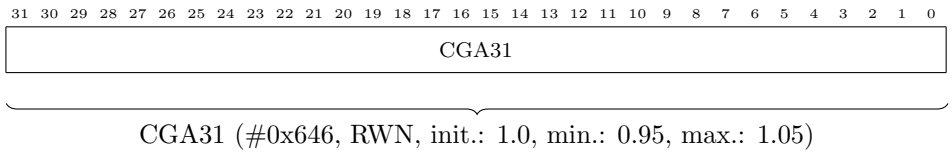
2.2.70 CGA30 (#0x645) - IN30 calibration gain register



CGA30 (#0x645, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA30 Calibration gain (Volts/Volts), see ADC calibration.

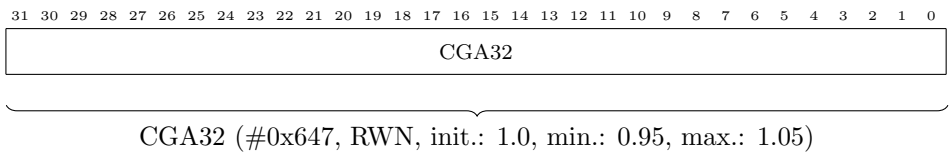
2.2.71 CGA31 (#0x646) - IN31 calibration gain register



CGA31 (#0x646, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA31 Calibration gain (Volts/Volts), see ADC calibration.

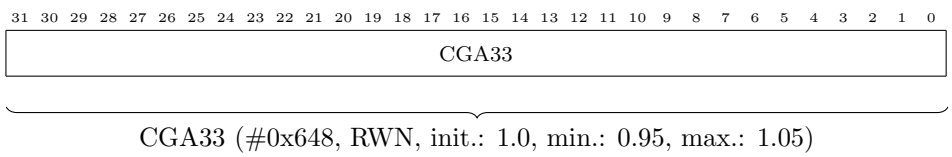
2.2.72 CGA32 (#0x647) - IN32 calibration gain register



CGA32 (#0x647, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA32 Calibration gain (Volts/Volts), see ADC calibration.

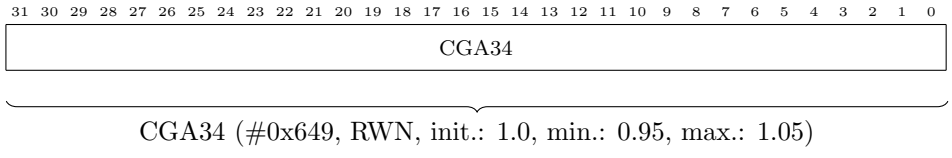
2.2.73 CGA33 (#0x648) - IN33 calibration gain register



CGA33 (#0x648, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA33 Calibration gain (Volts/Volts), see ADC calibration.

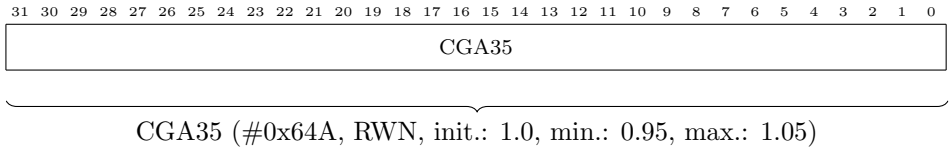
2.2.74 CGA34 (#0x649) - IN34 calibration gain register



CGA34 (#0x649, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA34 Calibration gain (Volts/Volts), see ADC calibration.

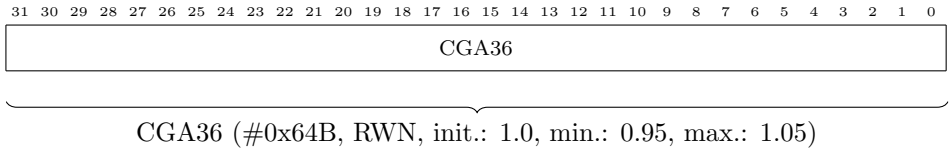
2.2.75 CGA35 (#0x64A) - IN35 calibration gain register



CGA35 (#0x64A, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA35 Calibration gain (Volts/Volts), see ADC calibration.

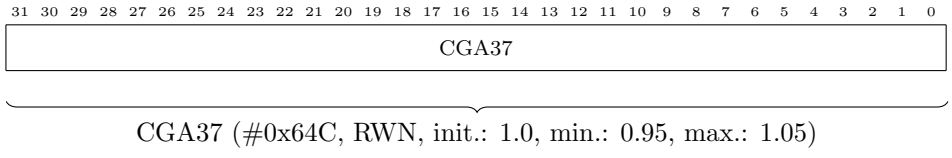
2.2.76 CGA36 (#0x64B) - IN36 calibration gain register



CGA36 (#0x64B, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA36 Calibration gain (Volts/Volts), see ADC calibration.

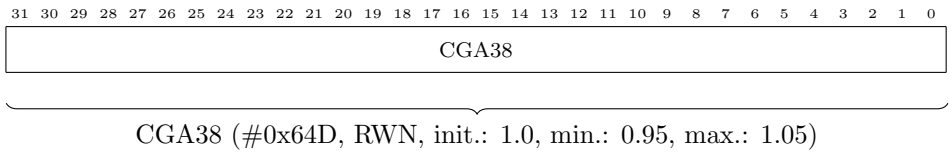
2.2.77 CGA37 (#0x64C) - IN37 calibration gain register



CGA37 (#0x64C, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA37 Calibration gain (Volts/Volts), see ADC calibration.

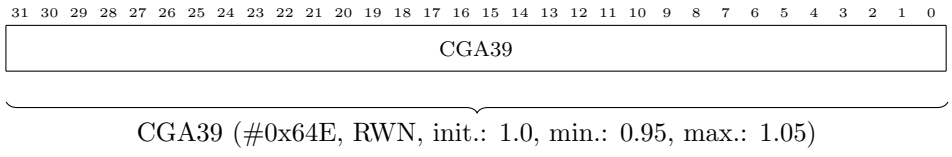
2.2.78 CGA38 (#0x64D) - IN38 calibration gain register



CGA38 (#0x64D, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA38 Calibration gain (Volts/Volts), see ADC calibration.

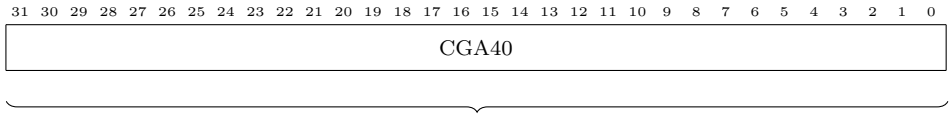
2.2.79 CGA39 (#0x64E) - IN39 calibration gain register



CGA39 (#0x64E, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA39 Calibration gain (Volts/Volts), see ADC calibration.

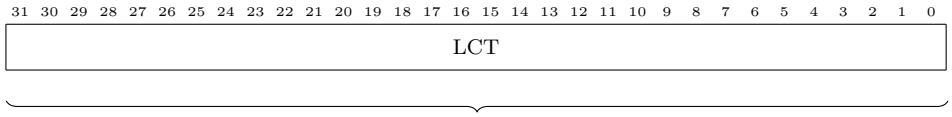
2.2.80 CGA40 (#0x64F) - IN40 calibration gain register



CGA40 (#0x64F, RWN, init.: 1.0, min.: 0.95, max.: 1.05)

[31:0] CGA40 — Calibration gain (Volts/Volts), see ADC calibration.

2.2.81 LCT (#0x650) - Last calibration time register

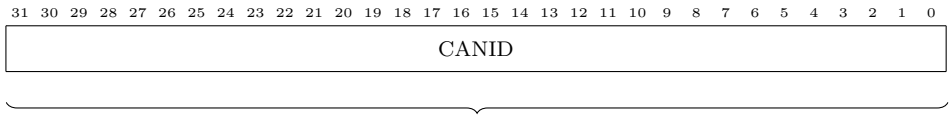


LCT (#0x650, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] LCT — Time of the last calibration (UCT) saved as Unix timestamp (compatible to C library time_t).

2.2.82 CANID (#0x651) - The node's CAN ID / address register

Writing to this register will change the node's address. The CANopen node will be stopped, initialized and then will automatically enter operational mode using the new address.



CANID (#0x651, RWN, init.: 60, min.: 60, max.: 69)

[31:0] CANID — The CAN node ID. See Setting the CAN node address for details.

3 ADC calibration

The calibration procedure is used to gain ADC accuracy better than 1 % over entire voltage range.

The calibration procedure requires presence of a calibration device with suitable accuracy class.

Each ADC channel calibration data consists of two registers: calibration offset voltage register (COF_X) and calibration gain register (CGA_X). These two registers are used when calculating the ADC readout for the particular ADC input using the following formula (X stands for ADC channel number):

$$\begin{cases} VIN_X = -\frac{4.096 \cdot \frac{ADC_X}{2^{16}} - 2.051}{0.195} \\ VCAL_X = VIN_X \cdot CGA_X + COF_X \end{cases} \tag{10}$$

where:

- ADC_X: the binary ADC readout
- VIN_X: the raw voltage readout
- CGA_X: calibration gain value
- COF_X: calibration offset voltage

- $VCAL_X$: the calibrated voltage readout

The calibration procedure is used to determine COF_X and CGA_X values for each ADC channel. It is based on a typical two point procedure:

1. A value of $V_X = 0$ V is applied and the measured voltage is stored as V_{X0}.
2. A value of $V_X =$ X V (where X is close to 10 V) is applied, the measured voltage is stored as V_{X10}.

The captured values are then used to determine OE_X (offset error) and GE_X (gain error) as follows:

$$\begin{cases} V_{X0} = OE_X \\ V_{X10} = GE_X \cdot X + OE_X \end{cases} \tag{11}$$

therefore:

$$\begin{cases} OE_X = V_{X0} \\ GE_X = \frac{V_{X10} - V_0}{X} \end{cases} \tag{12}$$

Knowing the offset and gain error for a channel, we can determine COF_X and CGA_X:

$$\begin{cases} V_X = GE_X \cdot VCAL_X + OE_X \\ V_X = GE_X \cdot VIN_X \cdot CGA_X + GE_X \cdot COF_X + OE_X \\ V_X \equiv VIN_X \Rightarrow GE_X \cdot CGA_X \equiv 1 \text{ and } GE_X \cdot COF_X + OE_X \equiv 0 \\ CGA_X \equiv \frac{1}{GE_X} \\ COF_X \equiv -\frac{OE_X}{GE_X} \end{cases} \tag{13}$$

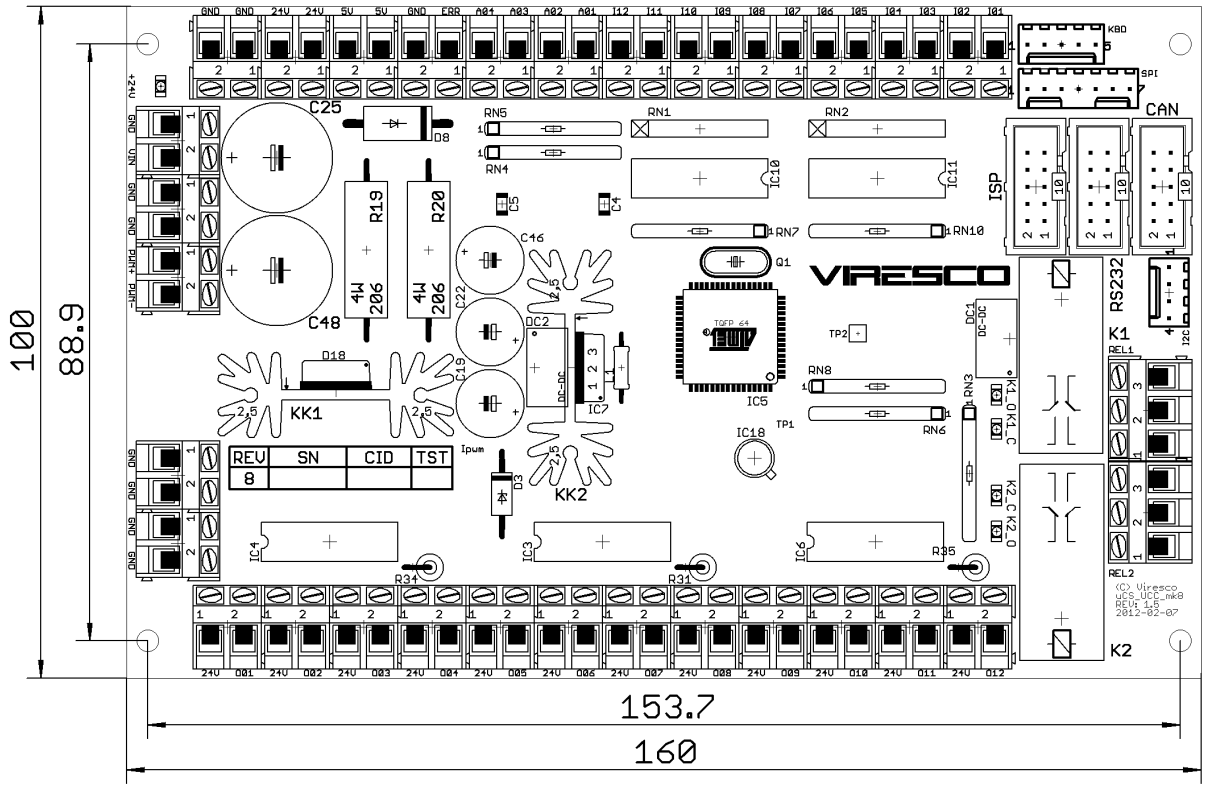
This procedure is repeated for all ADC channels and the calculated COF_X and CGA_X register values are written with the calculated values. The registers are then stored in non-volatile memory and used every time the ADC voltage is calculated.

VII. MATES-UCC-MK1 - universal CAN controller module

The UCC module is designed for industrial usage. It can serve not only as a test equipment, but also as a standalone control or measurement device.

The module is in Euro Card format and it is designed to work with the standard industrial 24 V actuators / sensors.

Figure 10: UCC PCB element side view



Unlike other MATES nodes, UCC is designed as OEM device and is available without PSU and case (PCB only).

1 Properties

UCC basic properties:

- 24 V power supply
- Twelve 24 V OC digital outputs with short-circuit protection
- Twelve 24 V digital IOs (open / ground suitable to be driven by contacts or open collector / open drain outputs)

- Four analog inputs 0÷5 V
- Panic stop input
- 5 A PWM driver
- 2 relays rating 8 A / 250 VAC
- 5 V supply output
- Power supply voltage measurement
- PWM driver current measurement
- SPI, I^2C, RS-232, CAN interfaces
- 4x1 keyboard connector
- LCD display connector (compatible with μOS_LCD)
- Digital temperature sensor

1.1 Outputs

1.1.1 Digital outputs O01-O12

There are 12 OC-type digital outputs. The sum of every of the three outputs groups (O01-O04, O05-O08 and O09-O12) can reach as much as 1 A[3] with any currents combination within a group (so essentially one output can drive a 1 A load if the remaining three outputs in a group are not used).

For each output, there is a corresponding 24 V connector, which along with the output itself can be used for powering a 24 V load such as relay coil. Every output has a "free wheel" rectifier diode connected to 24 V.

1.1.2 Digital outputs short circuit protection

The device has a short circuit protection. The threshold current is set to 1 A per group of four consequtive outputs.

! → The short circuit protection is a hardware/software mechanism, that is why its performance depends on proper behavior of the short circuit detection algorithm. Producer will not take responsibility for any damage that is caused by using software that is not factory programmed into the device.

1.1.3 Overview of the short circuit protection algorithm

These are the steps taken when a short circuit condition is encountered:

1. The short circuit condition is detected based on OVR signal which is common to all outputs (represents an OR of the short circuit conditions for all outputs).
2. The software enters service mode in which the main application has no access to the outputs.
3. The software starts a diagnostic loop in which the following actions are repeated over and over for each output:
   (a) turn on output X for about 5 μs;
   (b) check the OVR signal;
   (c) turn off output X;

   until the cause of the short circuit condition disappears.

[3] room temperature

4. After the short circuit condition disappears (which is concluded when turning on any of the outputs doesn't activate the OVR signal), a delay of 5 s is introduced.
5. The main application then resets the outputs to their current state.
6. The short circuit condition handling is completed.

! → A notable thing is that when the short circuit resolution algorithm is exercised, the main application has no access to the outputs. Particularly, note that a single output overload will deactivate (turn off) all outputs. Knowing that, the system has to be designed in a way in which the "off" state of all outputs is structurally safe.

! → The short circuit protection may not work when the ISP programmer is connected to the device..

Table 17: Digital outputs O01-O12

Parameter	Symbol	Min.	Typ.	Max.
Output voltage	U_O	1 V	-	24 V
Output current	I_O	-	0.25 A	1 A[4]
Turn on time	$t_{O_{on}}$	-	-	1 μs
Turn off time	$t_{O_{off}}$	-	-	1.5 μs

1.2 Inputs

1.2.1 Digital inputs I01-I12

The controller includes twelve digital inputs in industrial 24 V standard. Each input is pulled to +24 V (to the ground in the "A" version) using a weak resistor. This allows the inputs to be used with contact devices (switches, contactrons, relays) as well as OC outputs or push-pull outputs.

Table 18: Digital inputs I01-I12

Parameter	Symbol	Min.	Typ.	Max.
Maximum low state voltage	U_{DL}	-	-	7 V
Minimum high state voltage	U_{DH}	14 V	-	-
Pull-up resistance	R_{Dp}	-	3.3 kΩ	-
Input impedance	R_{Dse}	-	820 Ω	-

1.2.2 "Emergency stop" input

Aside standard logic inputs, the controller includes a special ERR input which, when open (disconnected from the ground), serves as an emergency stop indicator. The resistance of the circuit that is closing the ERR input has to be below 580 Ω. The ERR input probing voltage is 5 V. [5]

The emergency stop is by its nature a high sensitivity input. Because of that, even a small voltage glitch (about 2.5 V) can cause an unwanted activation. Having that in mind, a dedicated pair of wires should be used to connect the ERR / GND signals to the mushroom push button (or other device used as the emergency stop switch).

[4] absolute maximum possible only when just one output from a group is used, a group is four consequtive outputs (O01-O04, O05-O08, O09-O12)

[5] the parameters of the emergency stop input are chosen in a way, that the emergency stop switch can be connected in series with a red LED to indicate closed state (without additional serial resistor)

1.2.3 Analog inputs A01-A04

Inputs A01 to A04 are external analog voltage inputs. Each channel is equipped with active, Butterworth characteristic, unity gain filter.

Table 19 contains parameters of the analog inputs.

1.2.4 Analog channels A05-A06

There are two additional analog channels that are internally used in the UCC devices. These are:

- A05 which gives access to the power supply voltage (in Volts);
- A06 which gives access to the PWM output current (in Amperes).

1.2.5 Calibration of the analog inputs A01-A04

The device has a high quality voltage reference. This element, along with the progressive calibration procedure allows to gaining accuracy of ±1% ±10 mV in entire 0÷5 V operational range.

→ Attention: Analog inputs are not resilient to voltages outside 0÷5 V range.

Table 19: Analog inputs A01-A04

Parameter	Symbol	Min.	Typ.	Max.
Input voltage	V_A	0 V	-	5 V
Input resistance	$Z\|_{\omega=0}$	10 MΩ	-	-
Cutoff frequency	f_{-3dB}	-	1000 Hz	-
Filter slope	λ	-	-60 dB/dec	-
DC accuracy	Δ_V	-	-	0.1 % ±10 mV

1.3 Communication

The device is equipped with the following communication interfaces:

1. RS-232
2. HS-CAN
3. SPI
4. ISP
5. I^2C

1.3.1 RS-232

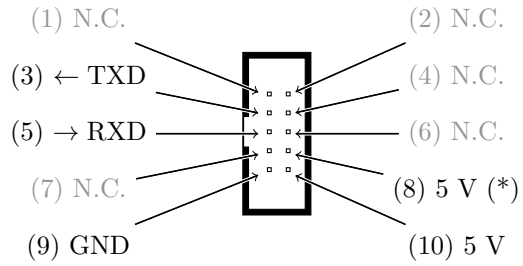
The serial interface is available at port designated as RS-232. This a standard IDC-10 socket. Pin placement was chosen in a way that a standard DSUB9-M plug can be crimped at the other end of the ribbon cable which can be connected to the PC computer or a 1 to 1 serial cable.

Only RXD and TXD signals are used. The hardware handshake signals are not provided.

Additionally, the 5 V supply voltage is routed to pin 10 and optionally to pin 8 of the socket. Pin 10 of the IDC-10 socket is not connected at the DSUB9 plug. On the other hand, pin 8 of the IDC-10 socket is connected to pin 9 of the DSUB9 plug. This pin is sometimes used to provide power for e.g. Bluetooth ↔ RS-232

converters. The presence of the 5 V supply at that pin makes it possible to provide power for such devices without any additional wiring.

Figure 11: RS-232 socket



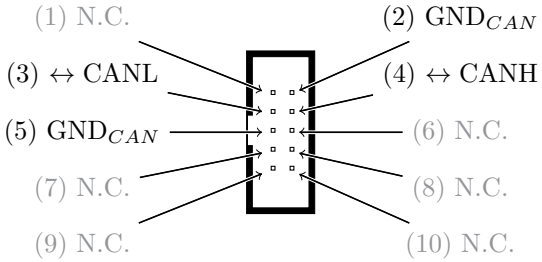
1.3.2 HS-CAN

The High Speed CAN is also available at IDC-10 socket. Pin placement is suitable for DSUB9 connector crimping (which then has a standard CAN signal mapping).

Note that using IDC-10 makes it possible to crimp three connectors at the same ribbon cable: DSUB9-M ↔ IDC-10 ↔ DSUB9-F. This arrangement allows easy connecting to a larger CAN bus.

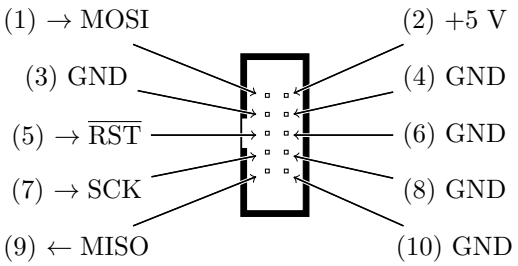
There is no place to mount the 120 Ω CAN terminating resistors. The terminator can be mounted in DSUB9 plug housing in the DSUB9 (bus) ↔ IDC-10 ↔ DSUB9 (terminator) pattern.

Figure 12: HS-CAN socket



1.3.3 ISP

Figure 13: ISP socket



2 Programming

2.1 Datagrams

2.1.1 Standard status - HEARTBEAT

7	6	5	4	3	2	1	0
STATUS							
HEARTBEAT (NMT = 700_h+ node number)							

[7:0] STATUS — Communication status of node, according to CANopen.

2.1.2 Digital inputs state - UCC-IN

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
I08	I07	I06	I05	I04	I03	I02	I01	KB4	KB3	KB2	KB1	I12	I11	I10	I09	RSVD						ERR	OVR
byte 0								byte 1								byte 2							
UCC-IN (PDO 1 TX = 180_h+ node number)																							

[0:7] IXX — The current state of the digital inputs (0 represents low electric state).

[8:11] KBX — The current state of the keyboard digital inputs (typically 0 denotes key pressed, this depends on the way the keyboard is connected).

[20:23] IXX — The current state of the digital inputs (0 represents low electric state).

[16:21] RSVD — Reserved for future use.

[22] ERR — The current state of the Error input (0 represents closed loop).

[23] OVR — The current state of the short circuit detector Overload signal (0 represents a short circuit condition).

2.1.3 Analogue input voltage - UCC-VIN

This datagram is used to get the input voltage for an individual ADC input. The provided voltage value is identical to the actual value at the input. The readout takes calibration data into account. You have to set the channel to read from before using this datagram (see ADC channel selection - UCC-CS).

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32 33 34 35 36 37 38 39	
CHANNEL	VALUE (LE)		
byte 0	byte 1	byte 2	byte 3	byte 4
UCC-VIN (PDO 2 TX = 280_h+ node number)			

[0:7] CHANNEL — ADC channel indicator (0 means ADC01, 5 means ADC06).

[8:39] VALUE — DAC voltage value [Volts], stored as IEEE-754 single precision floating point number in Little Endian.

2.1.4 Digital outputs state - UCC-OUT

This datagram is used by to set the value of the digital outputs of the device. The datagram value can be read back using remote frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
O08	O07	O06	O05	O04	O03	O02	O01	RSVD		K2	K1	O12	O11	O10	O09	RSVD		MODE		RSVD			
byte 0								byte 1								byte 2							
UCC-OUT (PDO 1 RX = 200_h+ node number)																							

[0:7] OXX — The current state of the digital outputs (0 represents low electric state).

[8:9] RSVD — Reserved for future use (must be set to zero).

[10:11] KX — The current state of the relays (0 means contacts in Normally Closed position).

[12:15] OXX — The current state of the digital outputs (0 represents low electric state).

[16:17] RSVD — Reserved for future use (must be set to zero).

[18:19] MODE — The outputs setting mode:

MODE	Name	Action
00_b	`SET`	OUT ← O01:K2
01_b	`AND`	OUT ← OUT & O01:K2
10_b	`OR`	OUT ← OUT | O01:K2
11_b	`XOR`	OUT ← OUT ^ O01:K2

! → When reading this datagram, the MODE field is always 0.

[20:23] RSVD — Reserved for future use (must be set to zero).

2.1.5 ADC channel selection - UCC-CS

The datagram is used to set the ADC channel number for the subsequent read operations. The datagram can be read back using remote frame.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CHANNEL								RSVD							
UCC-CS (PDO 2 RX = 300_h+ node number)															

[0:7] CHANNEL — ADC channel selector (0 means ADC01, 5 means ADC06).

[8:15] RSVD — Reserved for future use.

2.2 Programmable generators

The MATES-UCC-MK1 device can be programmed to produce simple waveforms on all 14 digital outputs. Each output is programmed using three registers:

- period register (e.g. PRD01);
- output rise time register (e.g. TRISE01);
- output fall time register (e.g. TFALL01).

All registers are programmed using values expressed in microseconds. Setting the period register to 0 µs disables the particular generator. When a generator is disabled, corresponding pin operates as a normal output. The generation for all channels is controlled using GENE bit in CR register. See the following figures for examples of registers settings.

2.2.1 Using regular IO along with generators

You can use regular IO functions when using generators. Channels that have their corresponding period register set to 0 can be controlled using normal procedures.

2.2.2 Manipulating output when generator is enabled

Setting value of an ouptut while the corresponding generator is enabled has no effect but it is not treated as an error. The value set during generator operation is saved in a shadow register and written to output as soon as the period register becomes 0.

2.2.3 Registers access vs the mates_setup_generator() API

There are few differences when programming generators directly using PRDx, TRISEx, TFALLx registers and using the dedicated `mates_setup_generator()` API.

When programming registers directly, no checks are performed and certain values will not produce a valid setup (like e.g. setting TRISEx above PRDx).

On the other hand, the `mates_setup_generator()` provides the additional functionality:

- Checks that all parameters have valid values in terms of generator resolution;
- Checks that period, rise time and fall time have valid values with respect of each other (rise time and fall time are no greater than period);
- If rise time is equal to period, sets the corresponding TRISEx register to 0;
- If fall time is equal to period, sets the corresponding TFALLx register to 0.

2.2.4 Examples

Figure 14: Example generator waveform

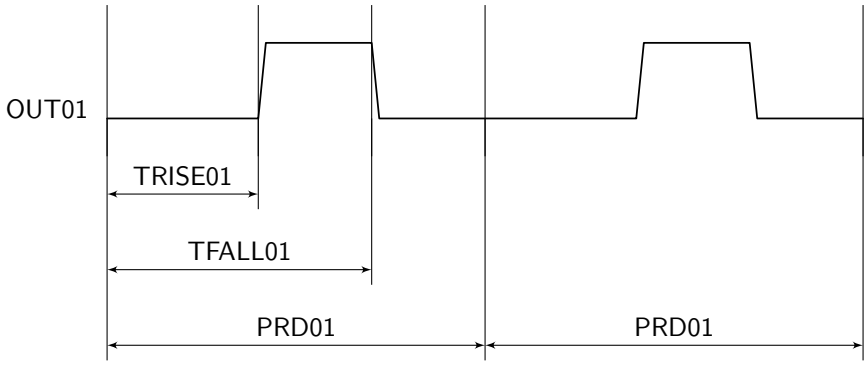


Figure 15: Negative pulse using TFALL < TRISE

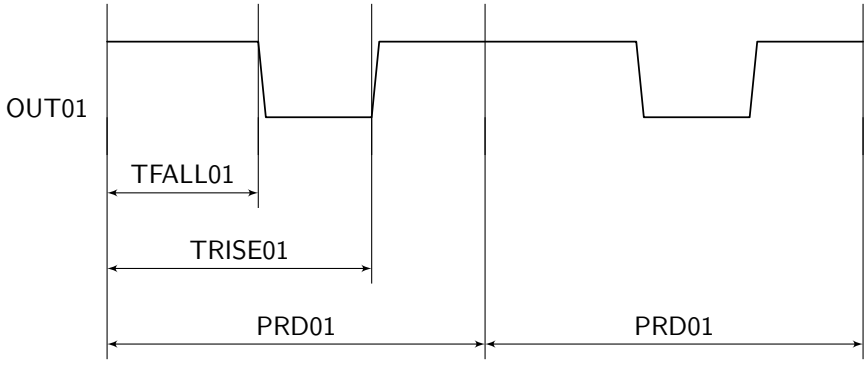
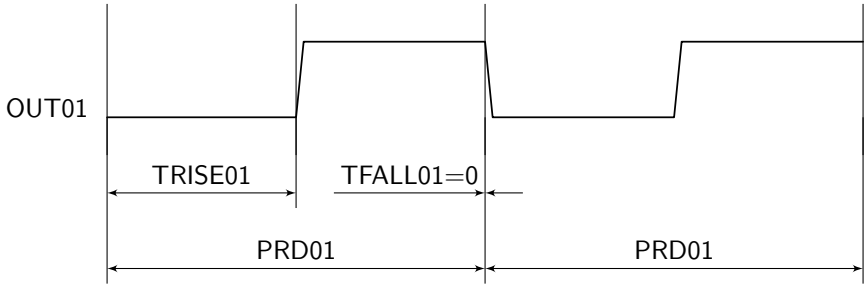
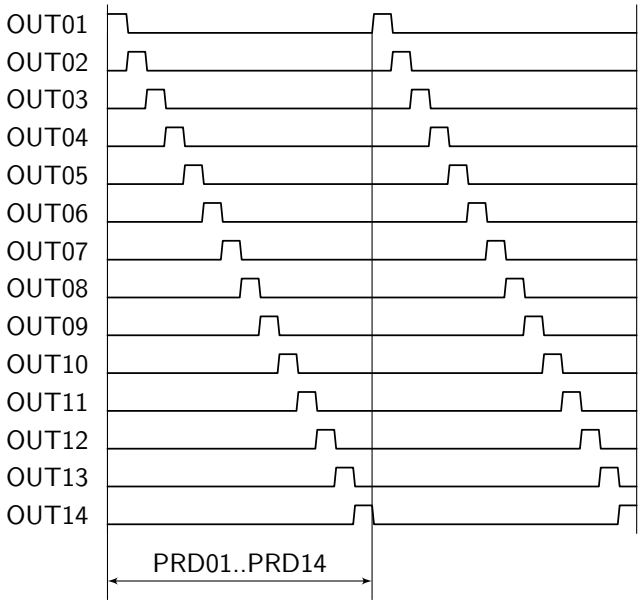


Figure 16: Signal fall at period end[a]



[a] Note that you cannot set TRISEx = PRDx or TFALLx = PRDx. These settings will not produce any transitions as the internal clock never reaches value of PRDx (it reaches value of PRDx - 1 and then rolls over to 0). Note however, that the `mates_setup_generator()` has a special handling for this case, see 2.2.3)

Figure 17: Emulating ring counter output



2.2.5 Generator resolution

For MATES-UCC-MK1, the time resolution is exactly:

$$T_q = 125\,\mu s \tag{14}$$

Writing a value that is not a multiple of T_q to any of the three generator registers results in an error. The T_q value may decrease in any future revisions of the hardware (the code backward compatibility will by preserved).

Using the specified T_q value, the highest frequency settings for a square (duty cycle is exactly 50%) waveform can be calculated as shown in table 20.

2.2.6 Using synchronized outputs

If two or more channels are programmed to have exactly the same period (or one period is a multiple of other), the channels run synchronously. You need to program

all channels before starting the generators[6] to use this function. If the generators are not stopped during programming, the respective time bases of generators are not guaranteed to be synchronized (clearing of the GENE bit resets all time bases to 0).

When channels are synchronized, the rise and fall times can be used to create some special patterns like Quadrature encoder waveform.

Figure 18: Synchronized channels

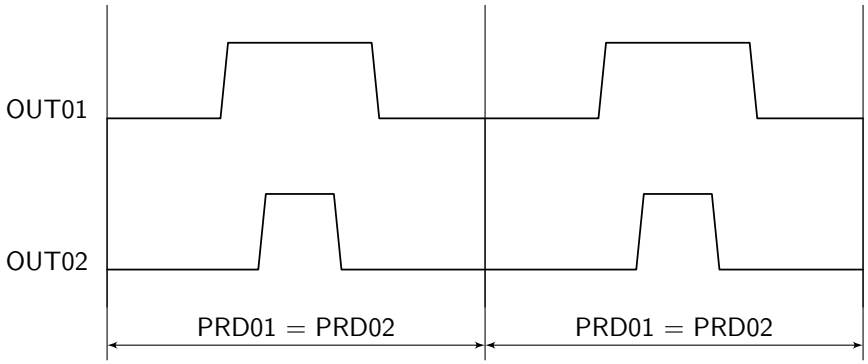
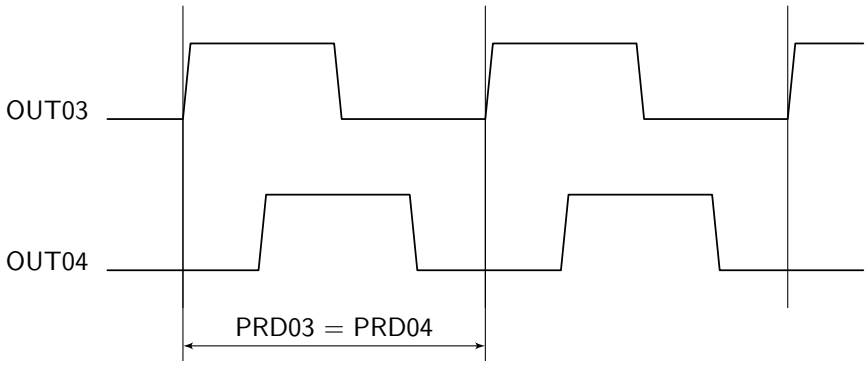


Figure 19: Quadrature encoder waveform



2.2.7 Generator latency

Due to programmatic nature of the signal generation, there is a small latency between actual voltage transitions on different pins even if all channels are programmed to change state at the same time (see OUT01 to OUT14 latency). This parameter, called T_{dly}, changes slightly for different generators configurations. The worst case scenario (all generators enabled, all transitions done in the same time, latency between OUT01 and OUT14) give the latency value of:

$$T_{dly}(max) = -1\,\mu s TBD \tag{15}$$

The best case scenario (delay between two consecutive channels):

$$T_{dly}(min) = -1\,\mu s TBD \tag{16}$$

[6] see GENE bit in CR register

Figure 20: OUT01 to OUT14 latency

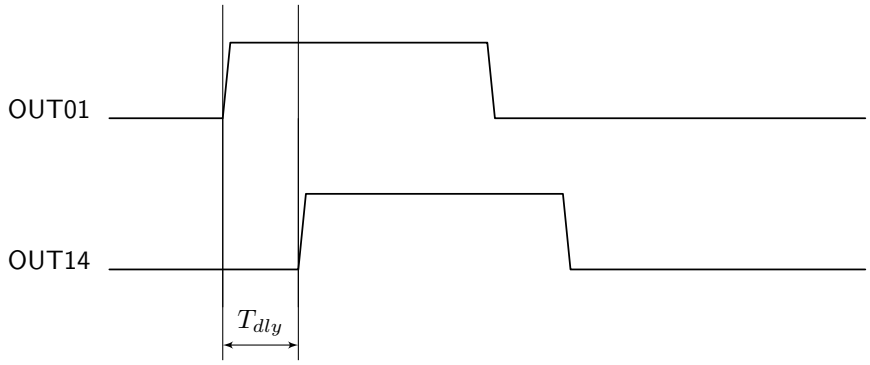
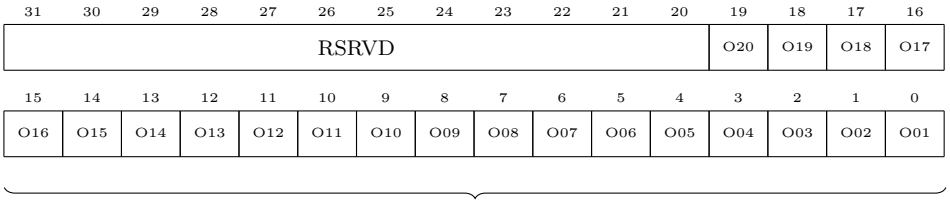


Table 20: Example square waveform parameters (duty = 50%)

Frequency [Hz]	PRDx = TFALLx	TRISEx
4000.000	250	125
2000.000	500	250
1333.333	750	375
1000.000	1000	500
800.000	1250	625
666.667	1500	750
571.429	1750	875
500.000	2000	1000
444.444	2250	1125
400.000	2500	1250
363.636	2750	1375
333.333	3000	1500
307.692	3250	1625
285.714	3500	1750
266.667	3750	1875
250.000	4000	2000
235.294	4250	2125
222.222	4500	2250
210.526	4750	2375
200.000	5000	2500
190.476	5250	2625
181.818	5500	2750
173.913	5750	2875
166.667	6000	3000
160.000	6250	3125
153.846	6500	3250
148.148	6750	3375
142.857	7000	3500
137.931	7250	3625
133.333	7500	3750
129.032	7750	3875
125.000	8000	4000
121.212	8250	4125
117.647	8500	4250
114.286	8750	4375
111.111	9000	4500
108.108	9250	4625
105.263	9500	4750
102.564	9750	4875
100.000	10000	5000

2.3 Registers

2.3.1 ODEF (#0x300) - Outputs default value register

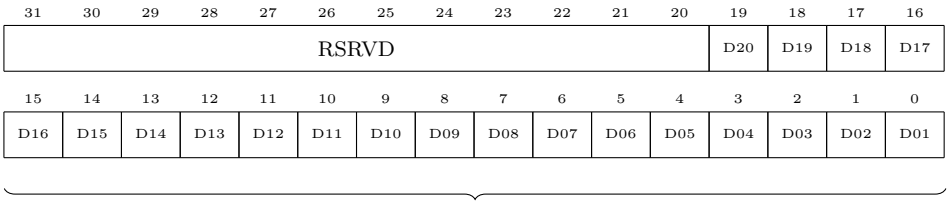


ODEF (#0x300, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFF)

[31:20]	RSRVD	Reserved, do not use.
[19]	O20	`OUT20` default value after startup.
[18]	O19	`OUT19` default value after startup.
[17]	O18	`OUT18` default value after startup.
[16]	O17	`OUT17` default value after startup.
[15]	O16	`OUT16` default value after startup.
[14]	O15	`OUT15` default value after startup.
[13]	O14	`OUT14` default value after startup.
[12]	O13	`OUT13` default value after startup.
[11]	O12	`OUT12` default value after startup.
[10]	O11	`OUT11` default value after startup.
[9]	O10	`OUT10` default value after startup.
[8]	O09	`OUT09` default value after startup.
[7]	O08	`OUT08` default value after startup.
[6]	O07	`OUT07` default value after startup.
[5]	O06	`OUT06` default value after startup.
[4]	O05	`OUT05` default value after startup.
[3]	O04	`OUT04` default value after startup.
[2]	O03	`OUT03` default value after startup.
[1]	O02	`OUT02` default value after startup.
[0]	O01	`OUT01` default value after startup.

2.3.2 ODIS (#0x301) - Outputs disable register

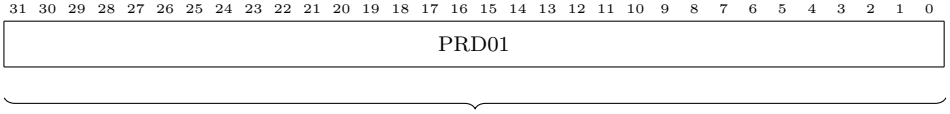
! → For MK1, all outputs are disabled at once, therefore the only valid value for this register is either 0x0 or 0xFFFFF. This behavior can change for future hardware revisions.



ODIS (#0x301, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFF)

[31:20]	RSRVD	Reserved, do not use.
[19]	D20	OUT20 disable bit (output enters high impedance state).
[18]	D19	OUT19 disable bit (output enters high impedance state).
[17]	D18	OUT18 disable bit (output enters high impedance state).
[16]	D17	OUT17 disable bit (output enters high impedance state).
[15]	D16	OUT16 disable bit (output enters high impedance state).
[14]	D15	OUT15 disable bit (output enters high impedance state).
[13]	D14	OUT14 disable bit (output enters high impedance state).
[12]	D13	OUT13 disable bit (output enters high impedance state).
[11]	D12	OUT12 disable bit (output enters high impedance state).
[10]	D11	OUT11 disable bit (output enters high impedance state).
[9]	D10	OUT10 disable bit (output enters high impedance state).
[8]	D09	OUT09 disable bit (output enters high impedance state).
[7]	D08	OUT08 disable bit (output enters high impedance state).
[6]	D07	OUT07 disable bit (output enters high impedance state).
[5]	D06	OUT06 disable bit (output enters high impedance state).
[4]	D05	OUT05 disable bit (output enters high impedance state).
[3]	D04	OUT04 disable bit (output enters high impedance state).
[2]	D03	OUT03 disable bit (output enters high impedance state).
[1]	D02	OUT02 disable bit (output enters high impedance state).
[0]	D01	OUT01 disable bit (output enters high impedance state).

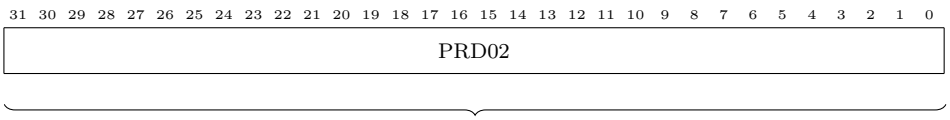
2.3.3 PRD01 (#0x302) - Channel 01 period register



PRD01 (#0x302, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0]	PRD01	Channel 01 waveform period (microseconds); set to 0 to disable the channel.

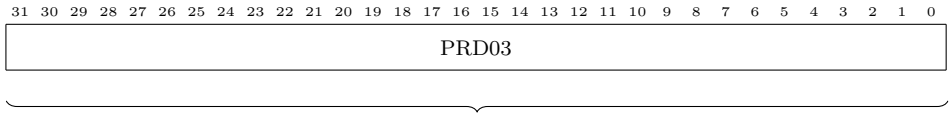
2.3.4 PRD02 (#0x303) - Channel 02 period register



PRD02 (#0x303, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD02 Channel 02 waveform period (microseconds); set to 0 to disable the channel.

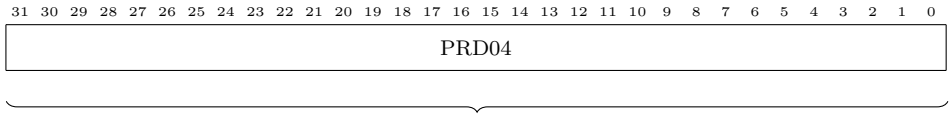
2.3.5 PRD03 (#0x304) - Channel 03 period register



PRD03 (#0x304, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD03 Channel 03 waveform period (microseconds); set to 0 to disable the channel.

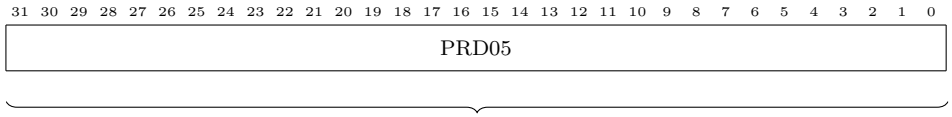
2.3.6 PRD04 (#0x305) - Channel 04 period register



PRD04 (#0x305, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD04 Channel 04 waveform period (microseconds); set to 0 to disable the channel.

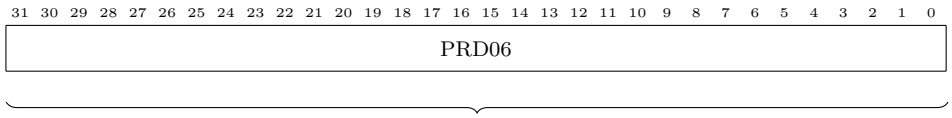
2.3.7 PRD05 (#0x306) - Channel 05 period register



PRD05 (#0x306, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD05 Channel 05 waveform period (microseconds); set to 0 to disable the channel.

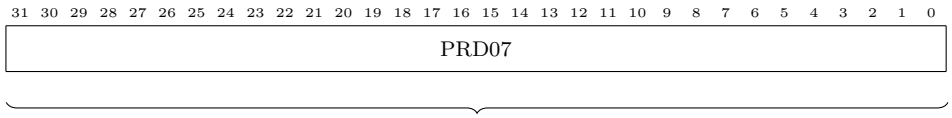
2.3.8 PRD06 (#0x307) - Channel 06 period register



PRD06 (#0x307, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD06 Channel 06 waveform period (microseconds); set to 0 to disable the channel.

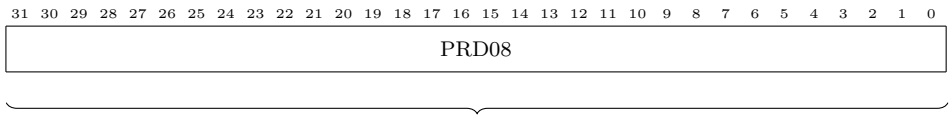
2.3.9 PRD07 (#0x308) - Channel 07 period register



PRD07 (#0x308, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD07 Channel 07 waveform period (microseconds); set to 0 to disable the channel.

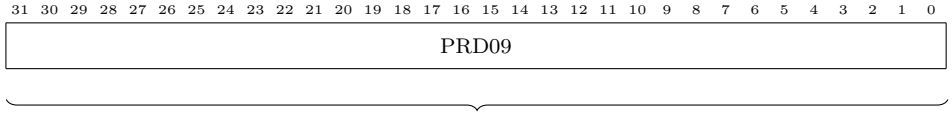
2.3.10 PRD08 (#0x309) - Channel 08 period register



PRD08 (#0x309, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD08 Channel 08 waveform period (microseconds); set to 0 to disable the channel.

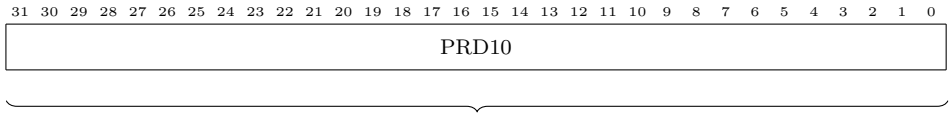
2.3.11 PRD09 (#0x30A) - Channel 09 period register



PRD09 (#0x30A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD09 — Channel 09 waveform period (microseconds); set to 0 to disable the channel.

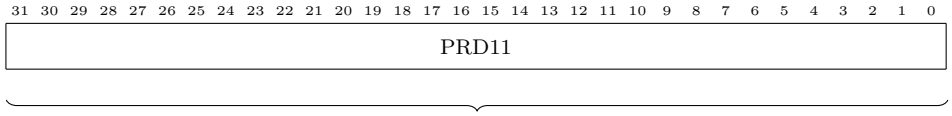
2.3.12 PRD10 (#0x30B) - Channel 10 period register



PRD10 (#0x30B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD10 — Channel 10 waveform period (microseconds); set to 0 to disable the channel.

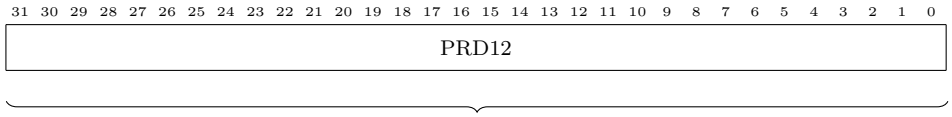
2.3.13 PRD11 (#0x30C) - Channel 11 period register



PRD11 (#0x30C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD11 — Channel 11 waveform period (microseconds); set to 0 to disable the channel.

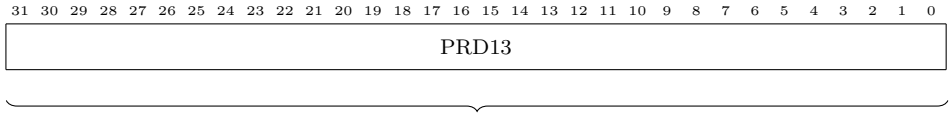
2.3.14 PRD12 (#0x30D) - Channel 12 period register



PRD12 (#0x30D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD12 — Channel 12 waveform period (microseconds); set to 0 to disable the channel.

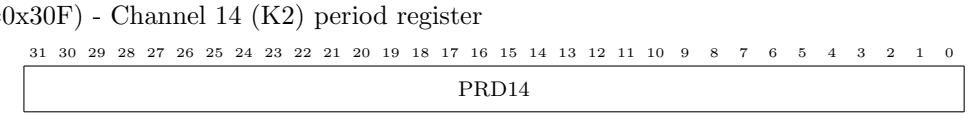
2.3.15 PRD13 (#0x30E) - Channel 13 (K1) period register



PRD13 (#0x30E, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] PRD13 — Channel 13 (relay K1) waveform period (microseconds); set to 0 to disable the channel.

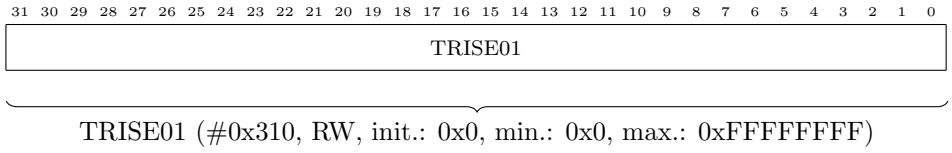
2.3.16 PRD14 (#0x30F) - Channel 14 (K2) period register



PRD14 (#0x30F, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

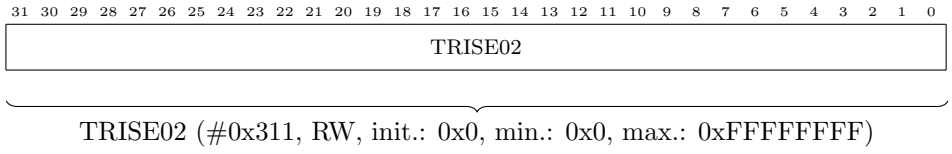
[31:0] PRD14 — Channel 13 (relay K1) waveform period (microseconds); set to 0 to disable the channel.

2.3.17 TRISE01 (#0x310) - Channel 01 rise time register



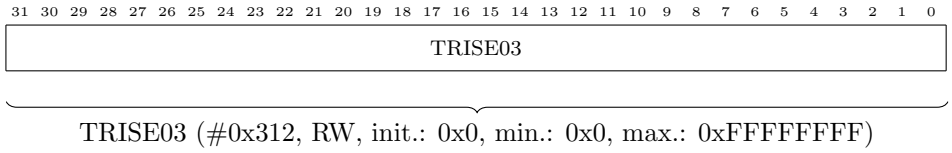
[31:0] TRISE01 Channel 01 signal rise time (microseconds).

2.3.18 TRISE02 (#0x311) - Channel 02 rise time register



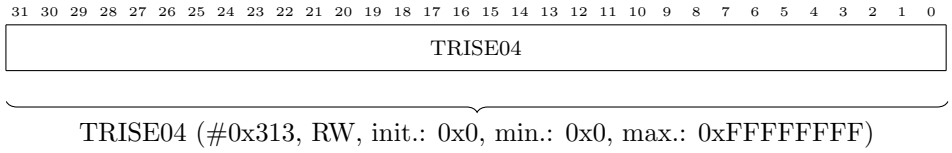
[31:0] TRISE02 Channel 02 signal rise time (microseconds).

2.3.19 TRISE03 (#0x312) - Channel 03 rise time register



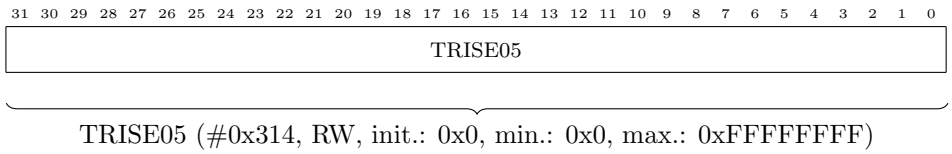
[31:0] TRISE03 Channel 03 signal rise time (microseconds).

2.3.20 TRISE04 (#0x313) - Channel 04 rise time register



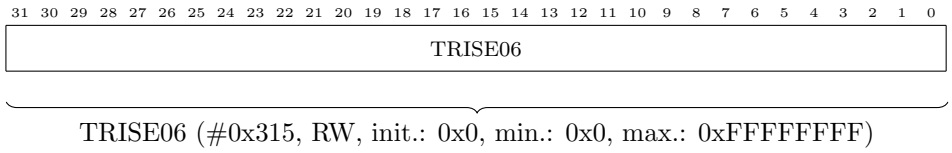
[31:0] TRISE04 Channel 04 signal rise time (microseconds).

2.3.21 TRISE05 (#0x314) - Channel 05 rise time register



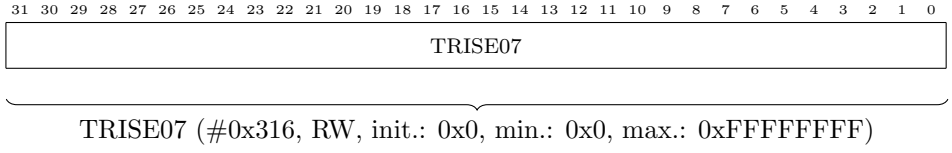
[31:0] TRISE05 Channel 05 signal rise time (microseconds).

2.3.22 TRISE06 (#0x315) - Channel 06 rise time register



[31:0] TRISE06 Channel 06 signal rise time (microseconds).

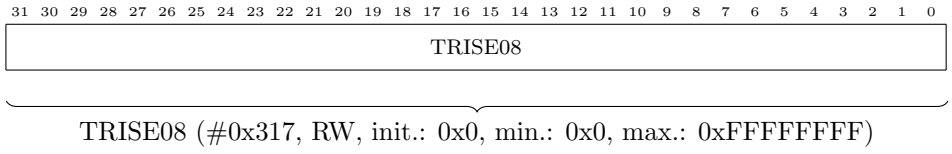
2.3.23 TRISE07 (#0x316) - Channel 07 rise time register



TRISE07 (#0x316, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE07 Channel 07 signal rise time (microseconds).

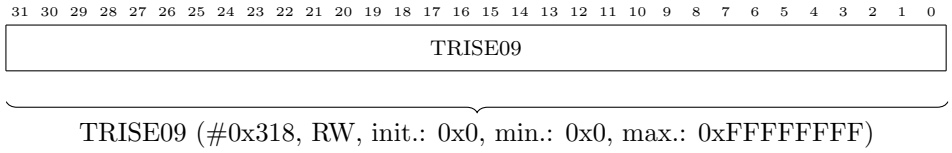
2.3.24 TRISE08 (#0x317) - Channel 08 rise time register



TRISE08 (#0x317, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE08 Channel 08 signal rise time (microseconds).

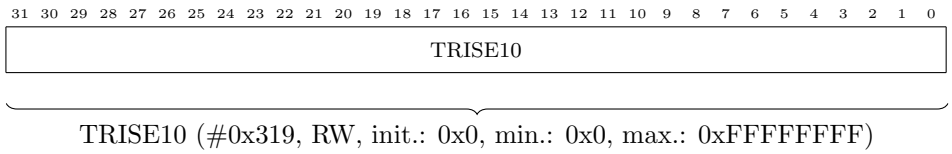
2.3.25 TRISE09 (#0x318) - Channel 09 rise time register



TRISE09 (#0x318, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE09 Channel 09 signal rise time (microseconds).

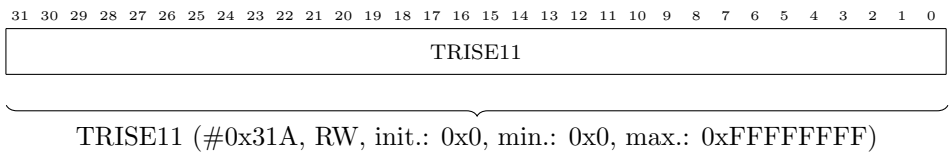
2.3.26 TRISE10 (#0x319) - Channel 10 rise time register



TRISE10 (#0x319, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE10 Channel 10 signal rise time (microseconds).

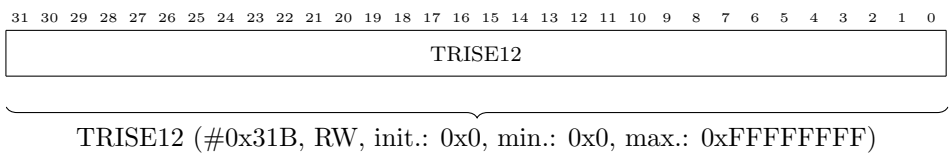
2.3.27 TRISE11 (#0x31A) - Channel 11 rise time register



TRISE11 (#0x31A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE11 Channel 11 signal rise time (microseconds).

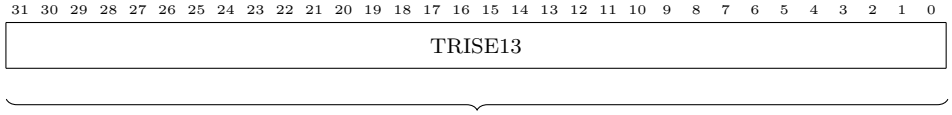
2.3.28 TRISE12 (#0x31B) - Channel 12 rise time register



TRISE12 (#0x31B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE12 Channel 12 signal rise time (microseconds).

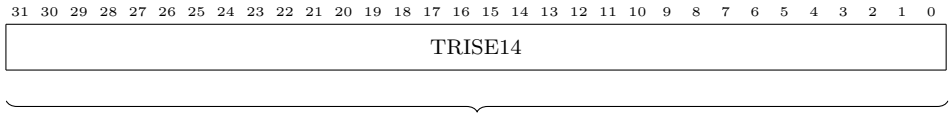
2.3.29 TRISE13 (#0x31C) - Channel 13 (K1) rise time register



TRISE13 (#0x31C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TRISE13 Channel 13 (relay K1) signal rise time (microseconds).

2.3.30 TRISE14 (#0x31D) - Channel 14 (K2) rise time register



TRISE14 (#0x31D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

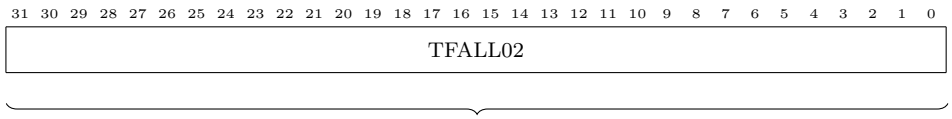
[31:0] TRISE14 Channel 14 (relay K2) signal rise time (microseconds).

2.3.31 TFALL01 (#0x31E) - Channel 01 fall time register

TFALL01 (#0x31E, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL01 Channel 01 signal fall time (microseconds).

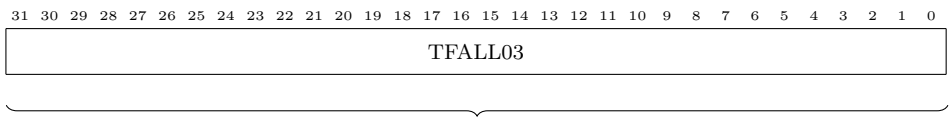
2.3.32 TFALL02 (#0x31F) - Channel 02 fall time register



TFALL02 (#0x31F, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL02 Channel 02 signal fall time (microseconds).

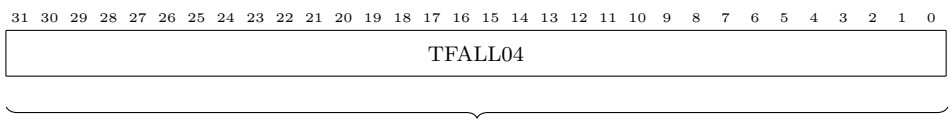
2.3.33 TFALL03 (#0x320) - Channel 03 fall time register



TFALL03 (#0x320, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL03 Channel 03 signal fall time (microseconds).

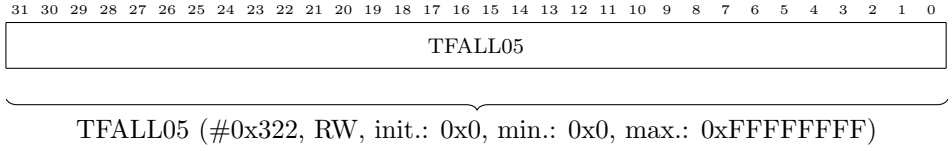
2.3.34 TFALL04 (#0x321) - Channel 04 fall time register



TFALL04 (#0x321, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL04 Channel 04 signal fall time (microseconds).

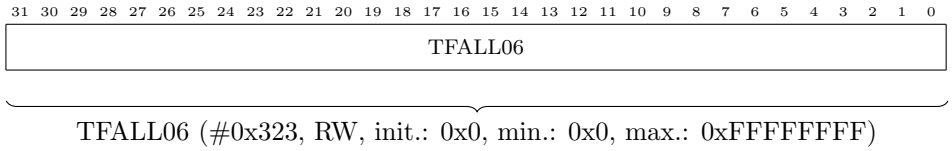
2.3.35 TFALL05 (#0x322) - Channel 05 fall time register



TFALL05 (#0x322, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL05 Channel 05 signal fall time (microseconds).

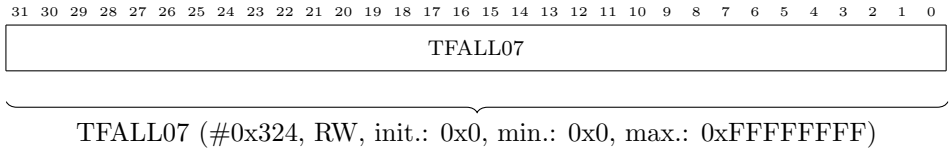
2.3.36 TFALL06 (#0x323) - Channel 06 fall time register



TFALL06 (#0x323, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL06 Channel 06 signal fall time (microseconds).

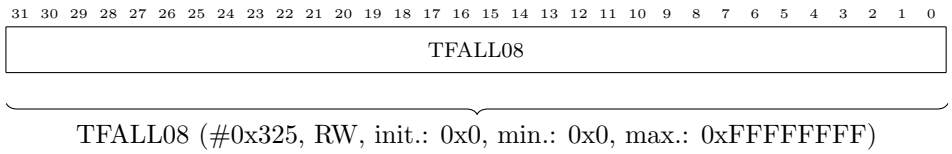
2.3.37 TFALL07 (#0x324) - Channel 07 fall time register



TFALL07 (#0x324, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL07 Channel 07 signal fall time (microseconds).

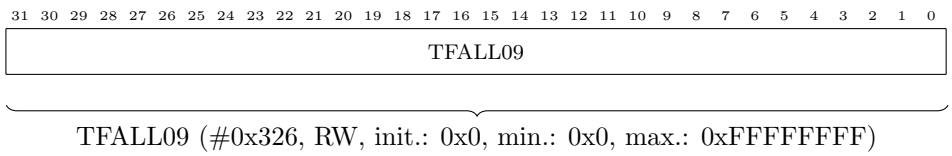
2.3.38 TFALL08 (#0x325) - Channel 08 fall time register



TFALL08 (#0x325, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL08 Channel 08 signal fall time (microseconds).

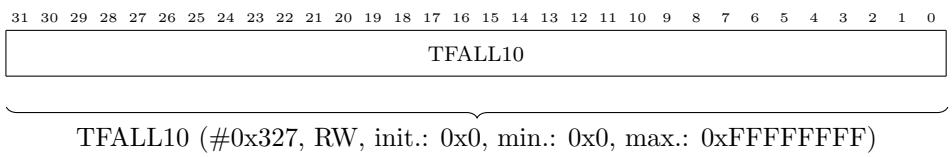
2.3.39 TFALL09 (#0x326) - Channel 09 fall time register



TFALL09 (#0x326, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL09 Channel 09 signal fall time (microseconds).

2.3.40 TFALL10 (#0x327) - Channel 10 fall time register



TFALL10 (#0x327, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL10 Channel 10 signal fall time (microseconds).

2.3.41 TFALL11 (#0x328) - Channel 11 fall time register

TFALL11 (#0x328, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL11 Channel 11 signal fall time (microseconds).

2.3.42 TFALL12 (#0x329) - Channel 12 fall time register

TFALL12 (#0x329, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL12 Channel 12 signal fall time (microseconds).

2.3.43 TFALL13 (#0x32A) - Channel 13 (K1) fall time register

TFALL13 (#0x32A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

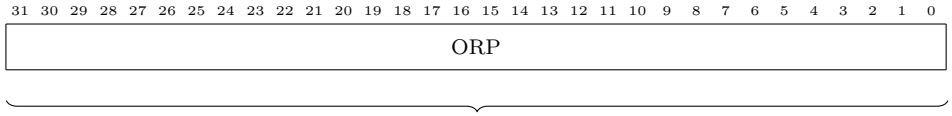
[31:0] TFALL13 Channel 13 (relay K1) signal fall time (microseconds).

2.3.44 TFALL14 (#0x32B) - Channel 14 (K2) fall time register

TFALL14 (#0x32B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] TFALL14 Channel 14 (relay K2) signal fall time (microseconds).

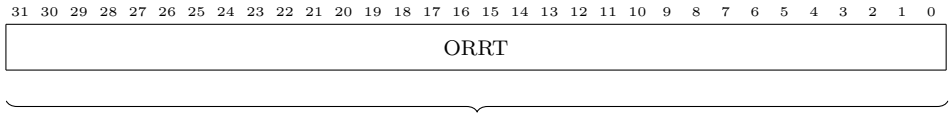
2.3.45 ORP (#0x32C) - Outputs restore period



ORP (#0x32C, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:0] ORP Outputs restore period (microseconds). When set to a non-zero value, defines the time after which all outputs are set to their default values if no output setting command is executed. Writing this register causes ORRT register to be reloaded.

2.3.46 ORRT (#0x32D) - Outptus restore remaining time register



ORRT (#0x32D, RO, init.: 0x0)

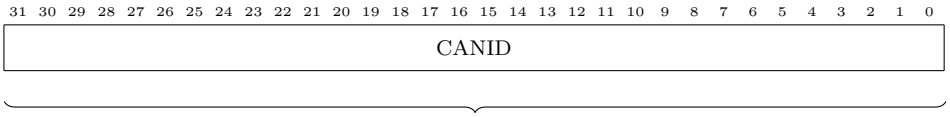
[31:0] ORRT Outputs restore remaining time (microseconds). Represents the time remaining before all outputs are restored to their default value. This register is reloaded with

the value of ORP each time the outputs state is being set (and also when ORP is written). When ORP is non-zero and the value of ORRT drops to zero, the following actions are performed:

- all signal generators are disabled by setting their PRDx registers to 0;
- the outputs are set to the value of ODEF register.

2.3.47 CANID (#0x32E) - The node's CAN ID / address register

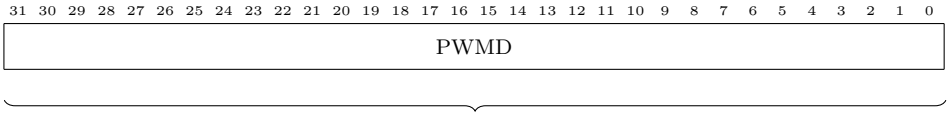
Writing to this register will change the node's address. The CANopen node will be stopped, initialized and then will automatically enter operational mode using the new address.



CANID (#0x32E, RWN, init.: 30, min.: 30, max.: 39)

[31:0] CANID — The CAN node ID. See Setting the CAN node address for details.

2.3.48 PWMD (#0x32F) - Pulse Width Modulator Duty

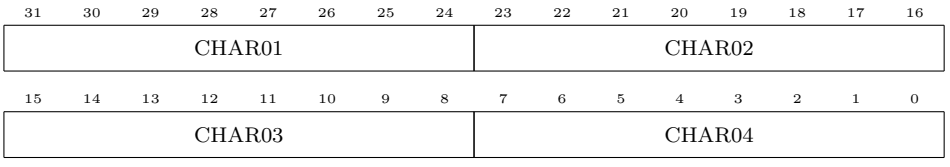


PWMD (#0x32F, RW, init.: 0.0, min.: 0.0, max.: 1.0)

[31:0] PWMD — PWM duty cycle normalized to [0.0:1.0] range. Values of exactly 0.0 and 1.0 will not produce any transitions.

2.3.49 ALCDB01 (#0x330) - Alphanumeric LCD display data register 1

Alphanumeric LCD display character data register.



ALCDB01 (#0x330, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR01 — Character #1 value.

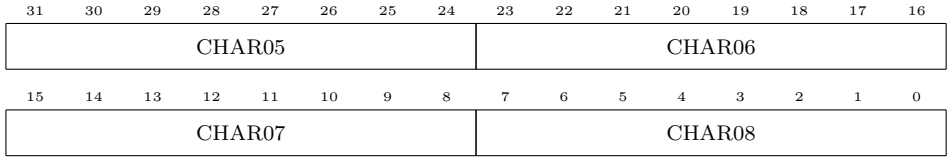
[23:16] CHAR02 — Character #2 value.

[15:8] CHAR03 — Character #3 value.

[7:0] CHAR04 — Character #4 value.

2.3.50 ALCDB02 (#0x331) - Alphanumeric LCD display data register 2

Alphanumeric LCD display character data register.



ALCDB02 (#0x331, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR05 Character #5 value.

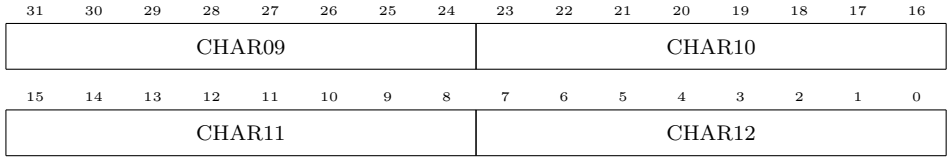
[23:16] CHAR06 Character #6 value.

[15:8] CHAR07 Character #7 value.

[7:0] CHAR08 Character #8 value.

2.3.51 ALCDB03 (#0x332) - Alphanumeric LCD display data register 3

Alphanumeric LCD display character data register.



ALCDB03 (#0x332, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR09 Character #9 value.

[23:16] CHAR10 Character #10 value.

[15:8] CHAR11 Character #11 value.

[7:0] CHAR12 Character #12 value.

2.3.52 ALCDB04 (#0x333) - Alphanumeric LCD display data register 4

Alphanumeric LCD display character data register.

ALCDB04 (#0x333, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR13 Character #13 value.

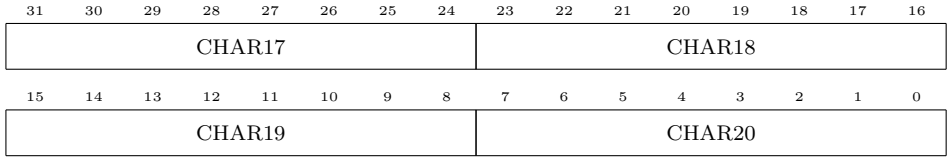
[23:16] CHAR14 Character #14 value.

[15:8] CHAR15 Character #15 value.

[7:0] CHAR16 Character #16 value.

2.3.53 ALCDB05 (#0x334) - Alphanumeric LCD display data register 5

Alphanumeric LCD display character data register.



ALCDB05 (#0x334, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR17 Character #17 value.

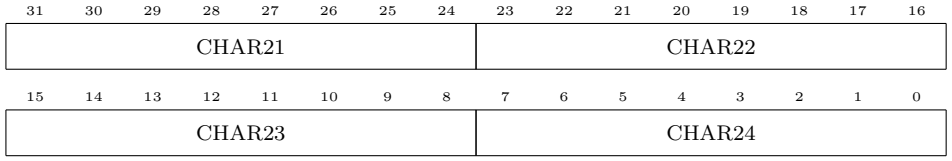
[23:16] CHAR18 Character #18 value.

[15:8] CHAR19 Character #19 value.

[7:0] CHAR20 Character #20 value.

2.3.54 ALCDB06 (#0x335) - Alphanumeric LCD display data register 6

Alphanumeric LCD display character data register.



ALCDB06 (#0x335, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR21 Character #21 value.

[23:16] CHAR22 Character #22 value.

[15:8] CHAR23 Character #23 value.

[7:0] CHAR24 Character #24 value.

2.3.55 ALCDB07 (#0x336) - Alphanumeric LCD display data register 7

Alphanumeric LCD display character data register.

ALCDB07 (#0x336, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24] CHAR25 Character #25 value.

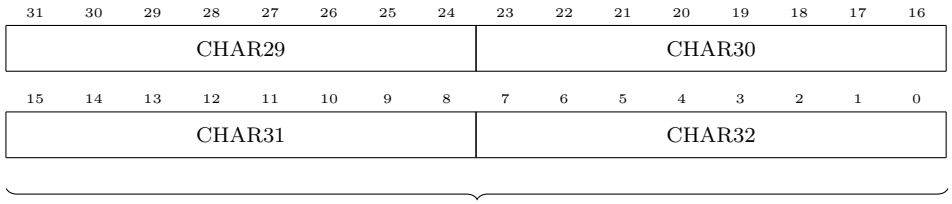
[23:16] CHAR26 Character #26 value.

[15:8] CHAR27 Character #27 value.

[7:0] CHAR28 Character #28 value.

2.3.56 ALCDB08 (#0x337) - Alphanumeric LCD display data register 8

Alphanumeric LCD display character data register.



ALCDB08 (#0x337, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

[31:24]	CHAR29	Character #29 value.
[23:16]	CHAR30	Character #30 value.
[15:8]	CHAR31	Character #31 value.
[7:0]	CHAR32	Character #32 value.

VIII. APPENDIX

A API reference

A.1 Introduction

This is MATES API documentation. The API is organized as follows:

- All API functions are available in single DLL file: *mates.dll*.
- The API functions are declared in *mates.h*.
- The MATES registers enumeration is defined in *mates_regs.h*.
- The Python MATES wrapper is provided by *viresco.mates*.
- The Python MATES registers definitions are in *viresco.mates.regs*.
- The C# wrapper class is defined in *Mates.cs* in *Viresco.Mates* namespace.
- The C# registers definitions are in *MatesRegs.cs*.

All mentioned files are included in the MATES installer (most recent version is *MATES-full-2.5.55.3587-setup.exe*). The location of files within the installation directory is described in the "Installation directory content" section of the MATES system manual: MATES.pdf.

A.2 Quick start

To link to the DLL you can use (example for MinGW port of GCC):

```
gcc main.c mates.dll -o main.exe
```

To run the example programs, you have to have the following DLLs present on your PATH (or in the current directory):

- mates.dll (2.7.0.0 or higher)
- uOS_monitor.dll (3.4.0.0 or higher)
- libxml2.dll (2.7.8)
- pcre3.dll (6.4.2194.685)
- zlib1.dll (1.2.5.0)

Besides the C DLL itself, the MATES API also has Python and C# wrappers. They are both using the same DLL.

The following Python modules need to be importable to use the Python wrapper:

- mates.py
- mates_regs.py

You can use the following code to add directory where these files are to Python search path:

```
1 import sys
2 sys.path.append("your directory")
3 import mates
4 ...
```

Alternatively, you can use one of the Wheel packages available in installation folder:

```
1 pip install "c:\Program Files (x86)\Viresco\MATES\bin\viresco.mates-1.0.0-py3-none-win32.whl"
```

And then simply import mates from viresco namespace:

```
1 from viresco import mates
2 ...
```

The Python wrapper was written against Python 2.7 32-bit and Python 3.6 32-bit.

To use the C# wrapper, add reference to the Viresco.Mates.dll assembly to your project and make sure the DLLs mentioned above are visible to your program. The Viresco.Mates.dll targets .NET 4.5.2.

All access methods were tested under Windows XP 32-bit, Windows Vista 32-bit, Windows 7 64-bit and Windows 10 64-bit.

Finally, the configuration file default.mon used in all examples need to be in the current directory (the one provided with installation is configured to use COM1 serial port, edit the file to fit your needs).

Before starting code development, you should be familiar with the MATES system manual: MATES.pdf. This document contains - among others - MATES registers descriptions.

A.3 Further reading

To view the API documentation go to:

- MATES DLL functions: mates.h
- MATES Python wrapper: viresco.mates
- MATES C# wrapper: Viresco.Mates

A.4 mates.h File Reference

MATES system access library.

Data Structures

- struct MATES_DESC_TYPE

Typedefs

- typedef MATES_DESC_TYPE * MATES_HANDLE

Enumerations

Functions

- void mates_set_verbosity (int level)

  Set the verbosity level.
- UOS_STATUS mates_monitor_version (char *buf, int max)

  Get the underlying uOS_monitor instance version.
- UOS_STATUS mates_monitor_path (char *buf, int max)

  Get the underlying uOS_monitor instance path.
- UOS_STATUS mates_dll_version (char *buf, int max)

  Get the mates.dll file version.
- UOS_STATUS mates_dll_path (char *buf, int max)

  Get the mates.dll file path.
- int mates_get_last_error (char *msg, int max)

  Get the lastly occurred error.
- int mates_get_last_error_ex (MATES_HANDLE h, char *msg, int max)

  Get the lastly occurred error for the specified MATES handle.
- void mates_print_errors (void)

  Print MATES error messages to standard output.
- void mates_print_errors_ex (MATES_HANDLE h)

  Print MATES error messages to standard output.
- void mates_clear_errors (void)

  Clear all error messages without printing.
- void mates_clear_errors_ex (MATES_HANDLE h)

  Clear all error messages without printing.
- MATES_HANDLE mates_open (char *conf_file_name, int addr)

  Open the MATES communication channel.
- MATES_HANDLE mates_open_buffer (char *conf_string, int addr)

  Open the MATES communication channel.
- UOS_STATUS mates_close (MATES_HANDLE h)

  Close the MATES communication channel.
- UOS_STATUS mates_discover_single_node (MATES_HANDLE h, int addr)

  Check if the specified node is present on the bus.
- UOS_STATUS mates_get_din_all (MATES_HANDLE h, int addr, int *value)

Get value of all digital inputs.

- UOS_STATUS mates_get_din (MATES_HANDLE h, int addr, int channel, int *value)

  Get value of an individual digital input.

- UOS_STATUS mates_set_dout_all (MATES_HANDLE h, int addr, int value)

  Set value of all digital outputs.

- UOS_STATUS mates_set_dout_all_ex (MATES_HANDLE h, int addr, int value, MATES_DIO_OUT_MODE_TYPE mode)

  Set value of all digital outputs using the specified mode.

- UOS_STATUS mates_set_dout (MATES_HANDLE h, int addr, int channel, int value)

  Set value of an individual digital output.

- UOS_STATUS mates_get_dout_all (MATES_HANDLE h, int addr, int *value)

  Get value of all digital outputs.

- UOS_STATUS mates_get_dout (MATES_HANDLE h, int addr, int channel, int *value)

  Get value of an individual digital output.

- UOS_STATUS mates_toggle_dout (MATES_HANDLE h, int addr, int channel)

  Toggle state of an individual digital output.

- UOS_STATUS mates_set_dac_raw (MATES_HANDLE h, int addr, int channel, int value)

  Set raw value of the DAC.

- UOS_STATUS mates_get_dac_raw (MATES_HANDLE h, int addr, int channel, int *value)

  Get the raw value of the DAC.

- UOS_STATUS mates_set_dac (MATES_HANDLE h, int addr, int channel, double value)

  Set the value of the DAC (Volts).

- UOS_STATUS mates_get_dac (MATES_HANDLE h, int addr, int channel, double *value)

  Get the value of the DAC (Volts).

- UOS_STATUS mates_get_adc (MATES_HANDLE h, int addr, int channel, double *value)

  Get the value of the ADC (Volts).

- UOS_STATUS mates_get_adc_raw (MATES_HANDLE h, int addr, int channel, int *value)

  Get the raw value of the ADC.

- UOS_STATUS mates_save_default (MATES_HANDLE h, int addr, int channel)

  Save the current value of an output channel as a default value.

- UOS_STATUS mates_set_register (MATES_HANDLE h, int addr, int reg, int value)

  Set regular (integral) MATES register.

- UOS_STATUS mates_set_register_f (MATES_HANDLE h, int addr, int reg, float value)

  Set floating point MATES register.
- UOS_STATUS mates_get_register (MATES_HANDLE h, int addr, int reg, int *value)

  Get regular (integral) MATES register.
- UOS_STATUS mates_get_register_f (MATES_HANDLE h, int addr, int reg, float *value)

  Get floating point MATES register.
- UOS_STATUS mates_get_register_sr (MATES_HANDLE h, int addr, int *value)

  Convenience function to get the status register SR.
- UOS_STATUS mates_get_register_fir (MATES_HANDLE h, int addr, int *value)

  Convenience function to get the firmware identification register FIR.
- UOS_STATUS mates_node_info (MATES_HANDLE h, int addr, char *buf, int max)

  Get node information as string.
- UOS_STATUS mates_enable_generators (MATES_HANDLE h, int addr, int enable)

  Enable signal generators for a node.
- UOS_STATUS mates_setup_generator (MATES_HANDLE h, int addr, int channel, unsigned int period, unsigned int rise_time, unsigned int fall_time)

  Setup signal generator for a digital channel.
- UOS_STATUS mates_lamp_test (MATES_HANDLE h, int addr)

  Execute the lamp test. See CR register documentation for details.
- UOS_STATUS mates_alcd_write (MATES_HANDLE h, int addr, int row, int col, char *text)

  Write text onto the alphanumeric LCD display.
- char * mates_status_name (UOS_STATUS code)

  Translate MATES status code into string representation.

A.4.1 Detailed Description

MATES system access library.

mates.h

```
Copyright (C) 2013-2018  Viresco
Lukasz Matecki  lukasz.matecki@viresco.pl
All rights reserved

Project:           MATES-DLL
Platform:          Windows XP SP3, Windows 7, Windows 8, Windows 10
Processor:         x86
```

Version $Revision: 4022 $

Date Created: 2013-07-19
Last modified: $Date:: 2019-01-21#$

Author Created by: Lukasz Matecki
Last modified by: $Author: Lukasz $

A.4.2 Typedef Documentation

typedef MATES_DESC_TYPE* MATES_HANDLE

Represents MATES connection handle.

A.4.3 Enumeration Type Documentation

enum UOS_STATUS

Generic status.

Enumerator *UOS_STATUS_ONE* : (1) When non-negative, UOS_STATUS may be used to indicate amount.

UOS_STATUS_OK : (0) No error or 0 when used to return amount.

UOS_STATUS_ERROR : (-1) Generic error.

UOS_STATUS_EINVAL : (-2) Invalid input value.

UOS_STATUS_ETIMEOUT : (-3) Timeout.

UOS_STATUS_ECORRUPT : (-4) Data corrupt.

UOS_STATUS_EBUSY : (-5) Resource is busy.

UOS_STATUS_ERANGE : (-6) Range error.

UOS_STATUS_EAGAIN : (-7) Could not complete operation, try again.

UOS_STATUS_ENODATA : (-8) There is no data available.

UOS_STATUS_ETOOBIG : (-9) There is too much data or too little room.

UOS_STATUS_EFULL : (-10) No more space available.

UOS_STATUS_ESTALE : (-11) The data is no longer valid.

enum MATES_NODE_NUMBER_TYPE

MATES node address enumeration. Used to distinguish node types and node specimens.

Enumerator *CANopen_node_min* : (1) Minimum node number.

mates_node_min_addr : (20) Minimum address for MATES node.

mates_dio3_mk1_1 : (20) MATES-DIO3-MK1, 1st device in the system.

mates_dio3_mk1_2 : (21) MATES-DIO3-MK1, 2nd device in the system.

mates_dio3_mk1_3 : (22) MATES-DIO3-MK1, 3rd device in the system.

mates_dio3_mk1_4 : (23) MATES-DIO3-MK1, 4th device in the system.

mates_dio3_mk1_5 : (24) MATES-DIO3-MK1, 5th device in the system.

mates_dio3_mk1_6 : (25) MATES-DIO3-MK1, 6th device in the system.

mates_dio3_mk1_7 : (26) MATES-DIO3-MK1, 7th device in the system.

mates_dio3_mk1_8 : (27) MATES-DIO3-MK1, 8th device in the system.

mates_dio3_mk1_9 : (28) MATES-DIO3-MK1, 9th device in the system.

mates_dio3_mk1_10 : (29) MATES-DIO3-MK1, 10th device in the system.

mates_ucc_mk1_1 : (30) MATES-UCC-MK1, 1st device in the system.

mates_ucc_mk1_2 : (31) MATES-UCC-MK1, 2nd device in the system.

mates_ucc_mk1_3 : (32) MATES-UCC-MK1, 3rd device in the system.

mates_ucc_mk1_4 : (33) MATES-UCC-MK1, 4th device in the system.

mates_ucc_mk1_5 : (34) MATES-UCC-MK1, 5th device in the system.

mates_ucc_mk1_6 : (35) MATES-UCC-MK1, 6th device in the system.

mates_ucc_mk1_7 : (36) MATES-UCC-MK1, 7th device in the system.

mates_ucc_mk1_8 : (37) MATES-UCC-MK1, 8th device in the system.

mates_ucc_mk1_9 : (38) MATES-UCC-MK1, 9th device in the system.

mates_ucc_mk1_10 : (39) MATES-UCC-MK1, 10th device in the system.

mates_diox_mk1_1 : (40) MATES-DIOX-MK1, 1st device in the system.

mates_diox_mk1_2 : (41) MATES-DIOX-MK1, 2nd device in the system.

mates_diox_mk1_3 : (42) MATES-DIOX-MK1, 3rd device in the system.

mates_diox_mk1_4 : (43) MATES-DIOX-MK1, 4th device in the system.

mates_diox_mk1_5 : (44) MATES-DIOX-MK1, 5th device in the system.

mates_diox_mk1_6 : (45) MATES-DIOX-MK1, 6th device in the system.

mates_diox_mk1_7 : (46) MATES-DIOX-MK1, 7th device in the system.

mates_diox_mk1_8 : (47) MATES-DIOX-MK1, 8th device in the system.

mates_diox_mk1_9 : (48) MATES-DIOX-MK1, 9th device in the system.

mates_diox_mk1_10 : (49) MATES-DIOX-MK1, 10th device in the system.

mates_dac5_mk1_1 : (50) MATES-DAC5-MK1, 1st device in the system.

mates_dac5_mk1_2 : (51) MATES-DAC5-MK1, 2nd device in the system.

mates_dac5_mk1_3 : (52) MATES-DAC5-MK1, 3rd device in the system.

mates_dac5_mk1_4 : (53) MATES-DAC5-MK1, 4th device in the system.

mates_dac5_mk1_5 : (54) MATES-DAC5-MK1, 5th device in the system.

mates_dac5_mk1_6 : (55) MATES-DAC5-MK1, 5th device in the system.

mates_dac5_mk1_7 : (56) MATES-DAC5-MK1, 6th device in the system.

mates_dac5_mk1_8 : (57) MATES-DAC5-MK1, 7th device in the system.

mates_dac5_mk1_9 : (58) MATES-DAC5-MK1, 8th device in the system.

mates_dac5_mk1_10 : (59) MATES-DAC5-MK1, 10th device in the system.

mates_adc5_mk1_1 : (60) MATES-ADC5-MK1, 1st device in the system.

mates_adc5_mk1_2 : (61) MATES-ADC5-MK1, 2nd device in the system.

mates_adc5_mk1_3 : (62) MATES-ADC5-MK1, 3rd device in the system.

mates_adc5_mk1_4 : (63) MATES-ADC5-MK1, 4th device in the system.

mates_adc5_mk1_5 : (64) MATES-ADC5-MK1, 5th device in the system.

mates_adc5_mk1_6 : (65) MATES-ADC5-MK1, 5th device in the system.

mates_adc5_mk1_7 : (66) MATES-ADC5-MK1, 6th device in the system.

mates_adc5_mk1_8 : (67) MATES-ADC5-MK1, 7th device in the system.

mates_adc5_mk1_9 : (68) MATES-ADC5-MK1, 8th device in the system.

mates_adc5_mk1_10 : (69) MATES-ADC5-MK1, 10th device in the system.

mates_node_max_addr : (69) Maximum address for MATES node.

CANopen_node_max : (127) Maximum node number.

enum MATES_OUTPUT_TYPE

MATES output channels enumeration.

Enumerator *MATES_OUT01* : (0) Output #1 - front panel designation: OUT01.

MATES_OUT02 : (1) Output #2 - front panel designation: OUT02.

MATES_OUT03 : (2) Output #3 - front panel designation: OUT03.

MATES_OUT04 : (3) Output #4 - front panel designation: OUT04.

MATES_OUT05 : (4) Output #5 - front panel designation: OUT05.

MATES_OUT06 : (5) Output #6 - front panel designation: OUT06.

MATES_OUT07 : (6) Output #7 - front panel designation: OUT07.

MATES_OUT08 : (7) Output #8 - front panel designation: OUT08.

MATES_OUT09 : (8) Output #9 - front panel designation: OUT09.

MATES_OUT10 : (9) Output #10 - front panel designation: OUT10.

MATES_OUT11 : (10) Output #11 - front panel designation: OUT11.

MATES_OUT12 : (11) Output #12 - front panel designation: OUT12.

MATES_OUT13 : (12) Output #13 - front panel designation: OUT13.

MATES_OUT14 : (13) Output #14 - front panel designation: OUT14.

MATES_OUT15 : (14) Output #15 - front panel designation: OUT15.

MATES_OUT16 : (15) Output #16 - front panel designation: OUT16.

MATES_OUT17 : (16) Output #17 - front panel designation: OUT17.

MATES_OUT18 : (17) Output #18 - front panel designation: OUT18.

MATES_OUT19 : (18) Output #19 - front panel designation: OUT19.

MATES_OUT20 : (19) Output #20 - front panel designation: OUT20.

MATES_OUT21 : (20) Output #21 - front panel designation: OUT21.

MATES_OUT22 : (21) Output #22 - front panel designation: OUT22.

MATES_OUT23 : (22) Output #23 - front panel designation: OUT23.

MATES_OUT24 : (23) Output #24 - front panel designation: OUT24.

MATES_OUT25 : (24) Output #25 - front panel designation: OUT25.

MATES_OUT26 : (25) Output #26 - front panel designation: OUT26.

MATES_OUT27 : (26) Output #27 - front panel designation: OUT27.

MATES_OUT28 : (27) Output #28 - front panel designation: OUT28.

MATES_OUT29 : (28) Output #29 - front panel designation: OUT29.

MATES_OUT30 : (29) Output #30 - front panel designation: OUT30.

MATES_OUT31 : (30) Output #31 - front panel designation: OUT31.

MATES_OUT32 : (31) Output #32 - front panel designation: OUT32.

MATES_OUT33 : (32) Output #33 - front panel designation: OUT33.

MATES_OUT34 : (33) Output #34 - front panel designation: OUT34.

MATES_OUT35 : (34) Output #35 - front panel designation: OUT35.

MATES_OUT36 : (35) Output #36 - front panel designation: OUT36.

MATES_OUT37 : (36) Output #37 - front panel designation: OUT37.

MATES_OUT38 : (37) Output #38 - front panel designation: OUT38.

MATES_OUT39 : (38) Output #39 - front panel designation: OUT39.

MATES_OUT40 : (39) Output #40 - front panel designation: OUT40.

enum MATES_INPUT_TYPE

Enumerator *MATES_IN01* : (0) Input #1 - front panel designation: IN01.

MATES_IN02 : (1) Input #2 - front panel designation: IN02.

MATES_IN03 : (2) Input #3 - front panel designation: IN03.

MATES_IN04 : (3) Input #4 - front panel designation: IN04.

MATES_IN05 : (4) Input #5 - front panel designation: IN05.

MATES_IN06 : (5) Input #6 - front panel designation: IN06.

MATES_IN07 : (6) Input #7 - front panel designation: IN07.

MATES_IN08 : (7) Input #8 - front panel designation: IN08.

MATES_IN09 : (8) Input #9 - front panel designation: IN09.

MATES_IN10 : (9) Input #10 - front panel designation: IN10.

MATES_IN11 : (10) Input #11 - front panel designation: IN11.

MATES_IN12 : (11) Input #12 - front panel designation: IN12.

MATES_IN13 : (12) Input #13 - front panel designation: IN13.

MATES_IN14 : (13) Input #14 - front panel designation: IN14.

MATES_IN15 : (14) Input #15 - front panel designation: IN15.

MATES_IN16 : (15) Input #16 - front panel designation: IN16.

MATES_IN17 : (16) Input #17 - front panel designation: IN17.

MATES_IN18 : (17) Input #18 - front panel designation: IN18.

MATES_IN19 : (18) Input #19 - front panel designation: IN19.

MATES_IN20 : (19) Input #20 - front panel designation: IN20.

MATES_IN21 : (20) Input #21 - front panel designation: IN21.

MATES_IN22 : (21) Input #22 - front panel designation: IN22.

MATES_IN23 : (22) Input #23 - front panel designation: IN23.

MATES_IN24 : (23) Input #24 - front panel designation: IN24.

MATES_IN25 : (24) Input #25 - front panel designation: IN25.

MATES_IN26 : (25) Input #26 - front panel designation: IN26.

MATES_IN27 : (26) Input #27 - front panel designation: IN27.

MATES_IN28 : (27) Input #28 - front panel designation: IN28.

MATES_IN29 : (28) Input #29 - front panel designation: IN29.

MATES_IN30 : (29) Input #30 - front panel designation: IN30.

MATES_IN31 : (30) Input #31 - front panel designation: IN31.

MATES_IN32 : (31) Input #32 - front panel designation: IN32.

MATES_IN33 : (32) Input #33 - front panel designation: IN33.

MATES_IN34 : (33) Input #34 - front panel designation: IN34.

MATES_IN35 : (34) Input #35 - front panel designation: IN35.

MATES_IN36 : (35) Input #36 - front panel designation: IN36.

MATES_IN37 : (36) Input #37 - front panel designation: IN37.

MATES_IN38 : (37) Input #38 - front panel designation: IN38.

MATES_IN39 : (38) Input #39 - front panel designation: IN39.

MATES_IN40 : (39) Input #40 - front panel designation: IN40.

enum MATES_DIO_OUT_MODE_TYPE

The outputs setting mode to be used with mates_set_dout_all_ex().

Enumerator *MATES_DIO_OUT_MODE_SET* : (0) SET mode.

MATES_DIO_OUT_MODE_AND : (1) AND mode.

MATES_DIO_OUT_MODE_OR : (2) OR mode.

MATES_DIO_OUT_MODE_XOR : (3) XOR mode.

A.4.4 Function Documentation

void mates_set_verbosity (
int *level*)

Set the verbosity level.

Parameters

`in`	*level*	The verbosity level: • 0 to disable verbose output; • 1 to print important messages; • 2 to print all messages.

Returns N/A.

All verbose messages are appended to the error log. They can be obtained the same way the error messages (using e.g. mates_get_last_error()).

UOS_STATUS mates_monitor_version (
char * *buf*,
int *max*)

Get the underlying *uOS_monitor* instance version.

Parameters

out	*buf*	Where to place the result; the version format is "X.X.X.X".
in	*max*	Amount of space in *buf*.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when there was not enough room in *buf*.
UOS_STATUS_ENODATA	when there was no alive handle to operate on.

Example usage (mates_test_20.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   char buf[100];
   if (mates_monitor_version(buf, sizeof(buf)) ==
      UOS_STATUS_OK)
   {
      printf("The monitor version used: %s\n", buf);
   }

   return (0);
}
```

Examples: mates_test_20.c.

UOS_STATUS mates_monitor_path (
char * *buf*,
int *max*)

Get the underlying *uOS_monitor* instance path.

Parameters

out	*buf*	Where to place the result.
in	*max*	Amount of space in *buf*.
in,out	-	

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when *buf* is NULL.

Since mates.dll 2.3.0.0

Example usage (mates_test_38.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   int status;
   char buf[260];
   if (mates_monitor_path(buf, sizeof(buf)) == UOS_STATUS_OK)
   {
      printf("Using monitor at '%s'\n", buf);
```

```
      return (0);
   }
   else
   {
      return (1);
   }
}
```

Examples: mates_test_38.c.

UOS_STATUS mates_dll_version (
char * *buf*,
int *max*)

Get the *mates.dll* file version.

Parameters

out	*buf*	Where to place the result; the version format is "X.X.X.X".
in	*max*	Amount of space in *buf*.

Returns The status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when there was not enough room in *buf*.
UOS_STATUS_ERROR	when the value cannot be established.

Example usage (mates_test_35.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   char buf[100];
   int success;
   if (mates_dll_version(buf, sizeof(buf)) == UOS_STATUS_OK)
   {
      printf("DLL version: %s\n", buf);
      return (0);
   }
   else
   {
      printf("Cannot get DLL version\n");
      return (1);
   }
}
```

Examples: mates_test_35.c.

UOS_STATUS mates_dll_path (
char * *buf*,
int *max*)

Get the *mates.dll* file path.

Parameters

out	*buf*	Where to place the result.
in	*max*	Amount of space in *buf*.
in,out	-	

Returns The status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when there was not enough room in *buf*.
UOS_STATUS_ERROR	when the value cannot be established.

Since mates.dll 2.3.0.0

Example usage (mates_test_37.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   int status;
   char buf[260];
   if (mates_dll_path(buf, sizeof(buf)) == UOS_STATUS_OK)
   {
      printf("Using DLL at '%s'\n", buf);
      return (0);
   }
   else
   {
      return (1);
   }
}
```

Examples: mates_test_37.c.

int mates_get_last_error (
char * *msg*,
int *max*)

Get the lastly occurred error.

Parameters

out	*msg*	Where to place the result.
in	*max*	Space available in *msg*.

Returns Status of the operation.

Return values

n	Number of characters that would have been written if there is enough room in *msg*.
0	on error.

Note This will get errors for any open MATES handle as well as errors not related to any handle (like handle creation errors). To get errors related only to a specific handle, use mates_get_last_error_ex().

Example usage (mates_test_21.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   char buf[100];
   int idx = 0;
   (void)mates_open(NULL, 0);
   while (mates_get_last_error(buf, sizeof(buf)) > 0)
   {
      /* Note: each message has a newline. */
      printf("MSG #%04d: %s", idx++, buf);
   }
}
```

Examples: mates_test_03.c, and mates_test_21.c.

int mates_get_last_error_ex (
MATES_HANDLE *h*,
char * *msg*,
int *max*)

Get the lastly occurred error for the specified MATES handle.

Parameters

in	*h*	The MATES communication handle.
out	*msg*	Where to place the result.
in	*max*	Amount of space in *msg*.

Returns Status of the operation.

Return values

Number	of characters that would have been written if there is enough room in *msg*.
0	on error.

See also mates_get_last_error().

Since mates.dll 2.0.0.0

Example usage (mates_test_34.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   char buf[100];
   int val;
   MATES_HANDLE h1 = mates_open("proxy.mon", 1);
   MATES_HANDLE h2 = mates_open("missing.mon", 0);
   mates_get_din_all(h1, mates_dio3_mk1_1, &val);
   int idx = 0;
   while (mates_get_last_error_ex(h1, buf, sizeof(buf)) > 0)
   {
      /* Note: each message has a newline. */
      printf("GET EX 1 #%04d: %s", idx++, buf);
   }
   /* Try empty handle. */
   while (mates_get_last_error_ex(NULL, buf, sizeof(buf)) > 0)
   {
      /* Note: each message has a newline. */
      printf("GET EX 2 #%04d: %s", idx++, buf);
   }
   /* Try another handle. */
   while (mates_get_last_error_ex(h2, buf, sizeof(buf)) > 0)
   {
      /* Note: each message has a newline. */
      printf("GET EX 3 #%04d: %s", idx++, buf);
   }
   mates_close(h1);
   mates_close(h2);

   return (0);
}
```

Examples: mates_test_34.c.

void mates_print_errors (
void)

Print MATES error messages to standard output.

Parameters

in	-	

Returns N/A.

Example usage (mates_test_18.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open(NULL, 0);
   if (h == NULL)
   {
      printf("MATES opening failed because:\n");
      mates_print_errors();
   }
   mates_close(h);
}
```

Examples: mates_test_08.c, mates_test_10.c, mates_test_18.c, mates_test_27.c, mates_test_28.c, mates_test_29.c, mates_test_33.c, and mates_test_41.c.

void mates_print_errors_ex (MATES_HANDLE *h*)

Print MATES error messages to standard output.

Parameters

in	*h*	The MATES communication handle.

Returns N/A.

Example usage (mates_test_36.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("default.mon", 0);
   mates_discover_single_node(h, 0);
   printf("MATES doscovery failed because:\n");
   mates_print_errors_ex(h);
   mates_close(h);
}
```

Examples: mates_test_36.c, and mates_test_39.c.

void mates_clear_errors (void)

Clear all error messages without printing.

Parameters

in	-	

Returns N/A.

Note The error buffer is never cleared internally (except the moment when handle is closed using mates_close()). If errors are expected, the user code should periodically clear them using this function, mates_clear_errors_ex(), mates_get_last_error() or mates_print_errors().

Deprecated This item is deprecated.

Use mates_clear_errors_ex() in new code.

Example usage (mates_test_19.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open(NULL, 0);
   int i;
   int sr;
   if (h == NULL)
   {
      for (i = 0; i < 100; i++)
      {
         /* Poll the status register. */
         if (mates_get_register_sr(h, mates_dio3_mk1_2, &sr) ==
      UOS_STATUS_OK)
         {
            printf("SR: %u\n", (unsigned)sr);
         }
         mates_clear_errors();
      }
   }
   mates_close(h);
}
```

Examples: mates_test_19.c.

void mates_clear_errors_ex (
MATES_HANDLE *h*)

Clear all error messages without printing.

Parameters

in	*h*	The MATES communication handle.

Returns N/A.

Note The error buffer is never cleared internally (except the moment when handle is closed using mates_close()). If errors are expected, the user code should periodically clear them using this function, mates_clear_errors(), mates_get_last_error() or mates_print_errors().

Remarks Use this function instead of mates_clear_errors() in new code.

Since mates.dll 1.2.0.0

MATES_HANDLE mates_open (
char * *conf_file_name*,
int *addr*)

Open the MATES communication channel.

Parameters

in	*conf_file_name*	*uOS_monitor* configuration file.
in	*addr*	The MATES device communication address to use (in the uOS_monitor communication network). Physical devices of the MK1 family have always addresses equal to 0. This has to have the same value as the corresponding *@map* attribute of the *.mon configuration file specified. The mates_proxy server assumes clients to connect using address of 1. Other values can be used when there is a proxy transport layer implemented between the physical device and mates.dll (like e.g. socket).

Returns MATES communication handle or NULL on error.

Example usage (mates_test_03.c):

```
#include <stdio.h>
#include <windows.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   if (h != NULL)
   {
      printf("Successfully opened MATES\n");
   }
   else
   {
      char buf[4096];
      while (mates_get_last_error(buf, sizeof(buf)) > 0)
      {
         printf(buf);
      }
   }

   /* Always close the handle. */
   mates_close(h);

   return (0);
}
```

Examples: mates_test_03.c, mates_test_04.c, mates_test_05.c, mates_test_06.c, mates_test_07.c, mates_test_08.c, mates_test_09.c, mates_test_10.c, mates_test_11.c, mates_test_12.c, mates_test_13.c, mates_test_14.c, mates_test_15.c, mates_test_16.c, mates_test_17.c, mates_test_18.c, mates_test_19.c, mates_test_21.c, mates_test_22.c, mates_test_23.c, mates_test_24.c, mates_test_25.c, mates_test_26.c, mates_test_27.c, mates_test_28.c, mates_test_29.c, mates_test_30.c, mates_test_31.c, mates_test_32.c, mates_test_33.c, mates_test_34.c, mates_test_36.c, mates_test_39.c, and mates_test_40.c.

MATES_HANDLE mates_open_buffer (char * *conf_string*, int *addr*)

Open the MATES communication channel.

Parameters

in	*conf_string*	*uOS_monitor* XML configuration string.
in	*addr*	The MATES device communication address to use (in the uOS_monitor communication network). Physical devices of the MK1 family have always addresses equal to 0. This has to have the same value as the corresponding *@map* attribute of the *.mon configuration file specified. The mates_proxy server assumes clients to connect using address of 1. Other values can be used when there is a proxy transport layer implemented between the physical device and mates.dll (like e.g. socket).

Returns MATES communication handle or NULL on error.

Since mates.dll 2.7.0.0

Example usage (mates_test_41.c):

```
#include <stdio.h>
#include "mates.h"

static char *conf = \
```

```
   "<?xml version=\"1.0\" encoding=\"utf-8\"?>"
   "<uOS_monitor_configuration>"
   " <local_address>1</local_address>"
   " <named_pipe map=\"1\">"
   "  <name>\\\\RAKIETA\\pipe\\MATES_PROXY_PIPE</name>"
   "  <timeout>5000</timeout>"
   " </named_pipe>"
   "</uOS_monitor_configuration>";

int main(void)
{
   int success = 0;
   char buf[0x200];
   int node = mates_ucc_mk1_1;
   MATES_HANDLE h = mates_open_buffer(conf, 1);
   if (mates_discover_single_node(h, node) ==
      UOS_STATUS_OK)
   {
      if (mates_node_info(h, node, buf, sizeof(buf)) ==
      UOS_STATUS_OK)
      {
         success++;
         printf("Found node #%d:\n", node);
         printf("%s\n", buf);
      }
      else
      {
         mates_print_errors();
      }
      mates_close(h);
      return (success == 1 ? 0 : -1);
   }
   else
   {
      return (0);
   }
}
```

Examples: mates_test_41.c.

UOS_STATUS mates_close (MATES_HANDLE *h*)

Close the MATES communication channel.

Parameters

in	*h*	The MATES communication handle.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	for invalid handle.
UOS_STATUS_ECORRUPT	when handle was already closed.

Example usage (mates_test_39.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   int success = 0;
   MATES_HANDLE h = mates_open("default.mon", 0);
   if (mates_discover_single_node(h, mates_ucc_mk1_1) ==
      UOS_STATUS_OK)
   {
      printf("Found UCC node.\n");
      mates_close(h);
      /* After closing of the handle it is not usable anymore. */
      if (mates_discover_single_node(h,
      mates_ucc_mk1_1) == UOS_STATUS_ECORRUPT)
      {
         success++;
      }
      if (mates_close(h) == UOS_STATUS_ECORRUPT)
```

```
        {
            success++;
        }
        mates_print_errors_ex(h);
        return (success == 2 ? 0 : -1);
    }
    else
    {
        return (0);
    }
}
```

This functions performs all necessary cleanup - closes the communication channels and frees any allocated resources.

Examples: mates_test_03.c, mates_test_04.c, mates_test_05.c, mates_test_06.c, mates_test_07.c, mates_test_08.c, mates_test_09.c, mates_test_10.c, mates_test_11.c, mates_test_12.c, mates_test_13.c, mates_test_14.c, mates_test_15.c, mates_test_16.c, mates_test_17.c, mates_test_18.c, mates_test_19.c, mates_test_22.c, mates_test_23.c, mates_test_24.c, mates_test_25.c, mates_test_26.c, mates_test_27.c, mates_test_28.c, mates_test_29.c, mates_test_30.c, mates_test_31.c, mates_test_32.c, mates_test_33.c, mates_test_34.c, mates_test_36.c, mates_test_39.c, mates_test_40.c, and mates_test_41.c.

UOS_STATUS mates_discover_single_node (MATES_HANDLE *h*, int *addr*)

Check if the specified node is present on the bus.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when the given node was found in the system.
UOS_STATUS_EINVAL	when *h* is invalid or *addr* is out of range.
UOS_STATUS_ETIMEOUT	when the node cannot be found.
UOS_STATUS_ERROR	on any other error.

Example usage (mates_test_04.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
    MATES_HANDLE h = mates_open("proxy.mon", 1);
    if (mates_discover_single_node(h, 20) ==
        UOS_STATUS_OK)
    {
        printf("Found node #20\n");
    }
    mates_close(h);
}
```

See also MATES manual

Examples: mates_test_04.c, mates_test_07.c, mates_test_08.c, mates_test_09.c, mates_test_10.c, mates_test_11.c, mates_test_22.c, mates_test_23.c, mates_test_24.c, mates_test_25.c, mates_test_26.c, mates_test_27.c, mates_test_28.c, mates_test_29.c, mates_test_30.c, mates_test_31.c, mates_test_32.c, mates_test_33.c, mates_test_36.c, mates_test_39.c, mates_test_40.c, and mates_test_41.c.

UOS_STATUS mates_get_din_all (
MATES_HANDLE *h*,
int *addr*,
int * *value*)

Get value of all digital inputs.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
out	*value*	Where to store the result, LSB is IN01.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_23.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   int value = 0;

   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_dio3_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Get the values of all digital inputs. */
   if (mates_get_din_all(h, mates_dio3_mk1_1, &value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully got all inputs: 0x%X\n", value);
   }
   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_23.c, mates_test_24.c, and mates_test_34.c.

UOS_STATUS mates_get_din (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
int * *value*)

Get value of an individual digital input.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DIN channel number [0,19].
out	*value*	Where to store the result.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_05.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int value;
   if (mates_get_din(h, mates_dio3_mk1_1,
      MATES_IN01, &value) == UOS_STATUS_OK)
   {
      printf("Channel #1: %d\n", value);
   }
   if (mates_get_din(h, mates_diox_mk1_1,
      MATES_IN01, &value) == UOS_STATUS_OK)
   {
      printf("Channel #1: %d\n", value);
   }
   mates_close(h);
}
```

Examples: mates_test_05.c, mates_test_06.c, and mates_test_32.c.

UOS_STATUS mates_set_dout_all (MATES_HANDLE *h*, int *addr*, int *value*)

Set value of all digital outputs.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*value*	The output value, LSB is OUT01.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_25.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   int value = 0xCAFE;
   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_diox_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Set the values of all digital outputs. */
   if (mates_set_dout_all(h, mates_diox_mk1_1, value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set all outputs: 0x%X\n", value);
   }
   value = 0;
   /* Read back. */
   if (mates_get_dout_all(h, mates_diox_mk1_1, &value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully got all outputs: 0x%X\n", value);
```

```
   }
   mates_close(h);
   return ((success == 2) ? 0 : -1);
}
```

Examples: mates_test_24.c, mates_test_25.c, mates_test_26.c, and mates_test_32.c.

UOS_STATUS mates_set_dout_all_ex (
MATES_HANDLE *h*,
int *addr*,
int *value*,
MATES_DIO_OUT_MODE_TYPE *mode*)

Set value of all digital outputs using the specified mode.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*value*	The output value, LSB is OUT01.
in	*mode*	Value setting mode: • 0: SET (OUT <- value) • 1: AND (OUT <- OUT & value) • 2: OR (OUT <- OUT | value) • 3: XOR (OUT <- OUT ^ value)

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_26.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   int value = 0xBACA;
   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_diox_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Set the values of all digital outputs. */
   if (mates_set_dout_all(h, mates_diox_mk1_1, value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set all outputs: 0x%X\n", value);
   }
   /* Now use AND mode to clear part of the outputs. */
   value = 0xFF0F;
   if (mates_set_dout_all_ex(h, mates_diox_mk1_1, value,
      MATES_DIO_OUT_MODE_AND) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully ANDed outputs: 0x%X\n", value);
   }
   if (mates_get_dout_all(h, mates_diox_mk1_1, &value) ==
      UOS_STATUS_OK &&
       value == 0xBA0A)
   {
      success++;
      printf("Successfully got all outputs: 0x%X\n", value);
   }
   mates_close(h);
```

```
   return ((success == 3) ? 0 : -1);
}
```

Examples: mates_test_26.c.

UOS_STATUS mates_set_dout (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
int *value*)

Set value of an individual digital output.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DOUT channel number [0,19].
in	*value*	The output value.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_06.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int value;
   if (mates_get_din(h, mates_dio3_mk1_1,
      MATES_IN01, &value) == UOS_STATUS_OK)
   {
      printf("Channel #1: %d\n", value);
   }

   /* Create a bridge between the input of one node and output of another node. */
   (void)mates_set_dout(h, mates_diox_mk1_1,
      MATES_OUT01, value);
   mates_close(h);
}
```

Examples: mates_test_06.c, mates_test_07.c, and mates_test_30.c.

UOS_STATUS mates_get_dout_all (
MATES_HANDLE *h*,
int *addr*,
int * *value*)

Get value of all digital outputs.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
out	*value*	Where to store the result, LSB is OUT01.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_24.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   unsigned int value = 0;

   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_dio3_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Get the values of all digital outputs. */
   if (mates_get_dout_all(h, mates_dio3_mk1_1, (int *)&value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully got all outputs: 0x%X\n", value);
   }
   /* Now negate. */
   value = (~value) & 0xFFFFF;
   if (mates_set_dout_all(h, mates_dio3_mk1_1, (int)value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set outputs: 0x%X\n", value);
   }
   /* Read back. */
   if (mates_get_dout_all(h, mates_dio3_mk1_1, (int *)&value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully got all outputs: 0x%X\n", value);
   }
   /* If we have loopback between inputs and outputs, we can also verify
      the outputs. */
   if (mates_get_din_all(h, mates_dio3_mk1_1, (int *)&value) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Successfully got all inputs: 0x%X\n", value);
   }
   mates_close(h);
   return ((success == 4) ? 0 : -1);
}
```

Examples: mates_test_24.c, mates_test_25.c, and mates_test_26.c.

UOS_STATUS mates_get_dout (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
int * *value*)

Get value of an individual digital output.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DOUT channel number [0,19].

out	*value*	Where to store the result.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_07.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   int value;

   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_dio3_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   if (mates_get_dout(h, mates_dio3_mk1_1,
      MATES_OUT20, &value) == UOS_STATUS_OK)
   {
      success++;
      printf("Channel #40 is set to: %d\n", value);
   }
   if (mates_set_dout(h, mates_dio3_mk1_1,
      MATES_OUT20, !value) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully toggled output\n");
   }
   mates_close(h);
   return ((success == 2) ? 0 : -1);
}
```

Examples: mates_test_07.c.

UOS_STATUS mates_toggle_dout (MATES_HANDLE *h*, int *addr*, int *channel*)

Toggle state of an individual digital output.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DOUT channel number [0,19].

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_08.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   if (mates_discover_single_node(h, mates_diox_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
```

```
      return (0);
   }
   if (mates_toggle_dout(h, mates_diox_mk1_1,
      MATES_OUT10) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully toggled output\n");
   }
   else
   {
      mates_print_errors();
   }
   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_08.c.

UOS_STATUS mates_set_dac_raw (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
int *value*)

Set raw value of the DAC.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DAC channel number [0,39].
in	*value*	The raw DAC binary code to write.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Note This function ignores the DAC calibration data.

Example usage (mates_test_09.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;

   if (mates_discover_single_node(h, mates_dac5_mk1_2) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }

   /* Set output to 1/10 of the full scale. */
   if (mates_set_dac_raw(h, mates_dac5_mk1_2,
      MATES_OUT30, 0xFFFF / 10) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set raw DAC value\n");
   }
   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_09.c, and mates_test_22.c.

UOS_STATUS mates_get_dac_raw (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
int * *value*)

Get the raw value of the DAC.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DAC channel number [0,39].
out	*value*	Where to place the result (raw DAC value).

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_22.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   int value = 0;

   if (mates_discover_single_node(h, mates_dac5_mk1_2) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }

   /* Set output to 1/10 of the full scale. */
   if (mates_set_dac_raw(h, mates_dac5_mk1_2,
      MATES_OUT30, 0xFFFF / 10) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set raw DAC value\n");
   }
   /* Read back the value. */
   value = 0;
   if (mates_get_dac_raw(h, mates_dac5_mk1_2,
      MATES_OUT30, &value) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully got raw DAC value: %d\n", value);
      if (value == (0xFFFF / 10))
      {
         success++;
      }
   }
   mates_close(h);
   return ((success == 3) ? 0 : -1);
}
```

Examples: mates_test_22.c.

UOS_STATUS mates_set_dac (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
double *value*)

Set the value of the DAC (Volts).

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DAC channel number [0,39].
in	*value*	The voltage value to set (Volts).

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_ERANGE	when written value is out of range.

Note This function takes calibration data into account.

Example usage (mates_test_10.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;

   if (mates_discover_single_node(h, mates_dac5_mk1_2) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }

   /* Set output to 2.5 V. */
   if (mates_set_dac(h, mates_dac5_mk1_2,
      MATES_OUT40, 2.5) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set DAC voltage value\n");
   }
   else
   {
      mates_print_errors();
   }
   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_10.c, and mates_test_11.c.

UOS_STATUS mates_get_dac (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
double * *value*)

Get the value of the DAC (Volts).

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The DAC channel number [0,39].
out	*value*	Where to place the result (voltage value in Volts).

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Warning When using mates_set_dac() and mates_get_dac() in a pair, the verified value may be slightly different than the set value. This is due to the fact that the value is internally converted to single precision floating point value so precision of the double precision argument can be lost. This behaviour may change in the future.

Example usage (mates_test_11.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   double val;
   if (mates_discover_single_node(h, mates_dac5_mk1_2) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Set output to 1.0 V. */
   if (mates_set_dac(h, mates_dac5_mk1_2,
      MATES_OUT40, 1.0) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully set DAC voltage value\n");
   }
   if (mates_get_dac(h, mates_dac5_mk1_2,
      MATES_OUT40, &val) == UOS_STATUS_OK)
   {
      success++;
      printf("Successfully get DAC voltage value\n");
   }
   if (val == 1.0)
   {
      success++;
   }
   mates_close(h);
   return ((success == 3) ? 0 : -1);
}
```

Examples: mates_test_11.c.

UOS_STATUS mates_get_adc (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
double * *value*)

Get the value of the ADC (Volts).

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The ADC channel number [0,39].
out	*value*	Where to place the result (voltage value in Volts).

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Note This function takes calibration data into account.

Example usage (mates_test_28.c):

```
#include <stdio.h>
```

```c
#include "mates.h"

int main(void)
{
   int success = 0;
   double val;
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_adc5_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   if (mates_get_adc(h, mates_adc5_mk1_1, 0, &val) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Value at channel IN 01: %f V\n", val);
   }
   else
   {
      mates_print_errors();
   }

   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_28.c.

UOS_STATUS mates_get_adc_raw (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
int * *value*)

Get the raw value of the ADC.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The ADC channel number [0,39].
out	*value*	Where to place the result (raw DAC value)).

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_29.c):

```c
#include <stdio.h>
#include "mates.h"

int main(void)
{
   int success = 0;
   int val;
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_adc5_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   if (mates_get_adc_raw(h, mates_adc5_mk1_1, 0, &val) ==
      UOS_STATUS_OK)
   {
      success++;
      printf("Binary value at channel IN 01: %d\n", val);
   }
   else
```

```
   {
      mates_print_errors();
   }

   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_29.c.

UOS_STATUS mates_save_default (
MATES_HANDLE *h*,
int *addr*,
int *channel*)

Save the current value of an output channel as a default value.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The output node channel number, [0,39] for DAC node, [0,19] for DIO3 or DIOX node.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when the node is not an output node or the channel number is out of range.

Example usage (mates_test_30.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;

   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_dio3_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Set value of output OUT01. */
   if (mates_set_dout(h, mates_dio3_mk1_1, 0, 1) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   /* Now save the new value as default. */
   if (mates_save_default(h, mates_dio3_mk1_1, 0) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   /* Repeat with default value of 0. */
   if (mates_set_dout(h, mates_dio3_mk1_1, 0, 0) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   if (mates_save_default(h, mates_dio3_mk1_1, 0) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   mates_close(h);
   return ((success == 4) ? 0 : -1);
}
```

Examples: mates_test_30.c.

UOS_STATUS mates_set_register (
MATES_HANDLE *h*,
int *addr*,
int *reg*,
int *value*)

Set regular (integral) MATES register.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*reg*	The register number to access (see MATES_REG_TYPE).
in	*value*	The register value.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when register number is invalid.
UOS_STATUS_ERANGE	when written value is out of range.

Example usage (mates_test_12.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   /* Outputs startup values for the fist 3.3 V node. */
   if (mates_set_register(h, mates_dio3_mk1_1,
      REG_MATES_DIO3_MK1_ODEF, 0xAAAAA) == UOS_STATUS_OK)
   {
      printf("Successfully set default values\n");
   }
   mates_close(h);
}
```

Examples: mates_test_12.c.

UOS_STATUS mates_set_register_f (
MATES_HANDLE *h*,
int *addr*,
int *reg*,
float *value*)

Set floating point MATES register.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*reg*	The register number to access (see MATES_REG_TYPE).

in	*value*	The register value.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when register number is invalid.
UOS_STATUS_ERANGE	when written value is out of range.

Example usage (mates_test_13.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   /* Set minimum value of 0.1 V for second DAC node at channel 1. */
   if (mates_set_register_f(h, mates_dac5_mk1_2,
      REG_MATES_DAC5_MK1_MIN01, 0.1) == UOS_STATUS_OK)
   {
      printf("Successfully set minimum value\n");
   }
   mates_close(h);
}
```

Examples: mates_test_13.c.

UOS_STATUS mates_get_register (
MATES_HANDLE *h*,
int *addr*,
int *reg*,
int * *value*)

Get regular (integral) MATES register.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*reg*	The register number to access (see MATES_REG_TYPE).
out	*value*	Where to place the result.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when register number is invalid.

Example usage (mates_test_14.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int bir;
   /* Get bootloader identification register. */
   if (mates_get_register(h, mates_dio3_mk1_2,
      REG_COMMON_BIR, &bir) == UOS_STATUS_OK)
   {
      printf("Successfully got BIR: 0x%X\n", bir);
   }
   mates_close(h);
}
```

Examples: mates_test_14.c.

UOS_STATUS mates_get_register_f (
MATES_HANDLE *h*,
int *addr*,
int *reg*,
float * *value*)

Get floating point MATES register.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*reg*	The register number to access (see MATES_REG_TYPE).
out	*value*	Where to place the result.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	when register number is invalid.

Example usage (mates_test_15.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   float offset;
   /* Get calibration offset for second DAC node, channel 1. */
   if (mates_get_register_f(h, mates_dac5_mk1_2,
      REG_MATES_DAC5_MK1_COF01, &offset) == UOS_STATUS_OK)
   {
      printf("Successfully got offset voltage: %f V\n", offset);
   }
   mates_close(h);
}
```

Examples: mates_test_15.c.

UOS_STATUS mates_get_register_sr (
MATES_HANDLE *h*,
int *addr*,
int * *value*)

Convenience function to get the status register SR.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
out	*value*	Where to place the result.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_16.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
```

```
   int sr;
   /* Get error status bit from SR. */
   if (mates_get_register_sr(h, mates_dio3_mk1_2, &sr) ==
      UOS_STATUS_OK)
   {
      printf("Error bit: %d\n", (sr & (1 << 1)) ? 1 : 0);
   }
   mates_close(h);
}
```

For registers definitions see MATES manual.

Examples: mates_test_16.c, and mates_test_19.c.

UOS_STATUS mates_get_register_fir (
MATES_HANDLE *h*,
int *addr*,
int * *value*)

Convenience function to get the firmware identification register FIR.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
out	*value*	Where to place the result.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_17.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int fir;
   /* Get firmware major version. */
   if (mates_get_register_fir(h, mates_dio3_mk1_2, &fir) ==
      UOS_STATUS_OK)
   {
      printf("Major firmware version: %d\n", (fir >> 24) & 0xFF);
   }
   mates_close(h);
}
```

For registers definitions see MATES manual.

Examples: mates_test_17.c.

UOS_STATUS mates_node_info (
MATES_HANDLE *h*,
int *addr*,
char * *buf*,
int *max*)

Get node information as string.

Parameters

in	*h*	The MATES communication handle.

in	*addr*	The node CAN address.
out	*buf*	Where to place the result.
in	*max*	The amount of space in *buf*.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Example usage (mates_test_27.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   char buf[0x200];
   int success = 0;
   MATES_HANDLE h = mates_open("proxy.mon", 1);

   /* Check if node is present. */
   if (mates_discover_single_node(h, 20) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   if (mates_node_info(h, 20, buf, sizeof(buf)) == UOS_STATUS_OK)
   {
      success++;
      printf("Found node #20:\n");
      printf(buf);
   }
   else
   {
      mates_print_errors();
   }

   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_27.c, and mates_test_41.c.

UOS_STATUS mates_enable_generators (
MATES_HANDLE *h*,
int *addr*,
int *enable*)

Enable signal generators for a node.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*enable*	The generators enable state.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Remarks Only MATES-DIO3-MK1 support signal generators.

Example usage (mates_test_31.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
```

```
   int success = 0;
   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_dio3_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Disable the generators and then enable them,
      this resets all time bases of the generators. */
   if (mates_enable_generators(h, mates_dio3_mk1_1, FALSE) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   if (mates_enable_generators(h, mates_dio3_mk1_1, TRUE) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   mates_close(h);
   return ((success == 2) ? 0 : -1);
}
```

Examples: mates_test_31.c, and mates_test_32.c.

UOS_STATUS mates_setup_generator (
MATES_HANDLE *h*,
int *addr*,
int *channel*,
unsigned int *period*,
unsigned int *rise_time*,
unsigned int *fall_time*)

Setup signal generator for a digital channel.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*channel*	The generator channel to setup [0,19].
in	*period*	The waveform period in microseconds.
in	*rise_time*	Signal rise time within the period in microseconds.
in	*fall_time*	Signal fall time within the period in microseconds.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_EINVAL	if • *period*, *rise_time* or *fall_time* are not multiple of 125 us; • *rise_time* or *fall_time* are greater than *period*; • *addr* points to a node that doesn't support digital signal generators; • *channel* is out of range.

Note *period*, *rise_time* and *fall_time* have to be a multiple of 125 us.

Remarks Signal generators are supported by MATES-DIO3-MK1 and MATES-UCC-MK1 nodes.

Example usage (mates_test_32.c):

```c
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   int i1, i2;
   /* Check if node is present. */
   if (mates_discover_single_node(h, mates_dio3_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_close(h);
      return (0);
   }
   /* Generate quadrature waveform on OUT01 and OUT02 at 1 Hz. */
   mates_set_dout_all(h, mates_dio3_mk1_1, 0);
   mates_enable_generators(h, mates_dio3_mk1_1, FALSE);
   mates_setup_generator(h, mates_dio3_mk1_1, 0, 1000000, 0, 500000);
   mates_setup_generator(h, mates_dio3_mk1_1, 1, 1000000, 250000, 7500
      00);
   mates_enable_generators(h, mates_dio3_mk1_1, TRUE);

   /* Assume we have outputs to inputs loopback present. */
   if (mates_get_din(h, mates_dio3_mk1_1, 0, &i1) ==
      UOS_STATUS_OK &&
       mates_get_din(h, mates_dio3_mk1_1, 1, &i2) ==
      UOS_STATUS_OK &&
       i1 == 1 && i2 == 0)
   {
      success++;
   }
   printf("i1 = %d, i2 = %d\n", i1, i2);
   Sleep(260);
   if (mates_get_din(h, mates_dio3_mk1_1, 0, &i1) ==
      UOS_STATUS_OK &&
       mates_get_din(h, mates_dio3_mk1_1, 1, &i2) ==
      UOS_STATUS_OK &&
       i1 == 1 && i2 == 1)
   {
      success++;
   }
   printf("i1 = %d, i2 = %d\n", i1, i2);
   Sleep(250);
   if (mates_get_din(h, mates_dio3_mk1_1, 0, &i1) ==
      UOS_STATUS_OK &&
       mates_get_din(h, mates_dio3_mk1_1, 1, &i2) ==
      UOS_STATUS_OK &&
       i1 == 0 && i2 == 1)
   {
      success++;
   }
   printf("i1 = %d, i2 = %d\n", i1, i2);
   Sleep(250);
   if (mates_get_din(h, mates_dio3_mk1_1, 0, &i1) ==
      UOS_STATUS_OK &&
       mates_get_din(h, mates_dio3_mk1_1, 1, &i2) ==
      UOS_STATUS_OK &&
       i1 == 0 && i2 == 0)
   {
      success++;
   }
   printf("i1 = %d, i2 = %d\n", i1, i2);

   /* Disable generators. */
   mates_setup_generator(h, mates_dio3_mk1_1, 0, 0, 0, 0);
   mates_setup_generator(h, mates_dio3_mk1_1, 1, 0, 0, 0);
   mates_enable_generators(h, mates_dio3_mk1_1, FALSE);
   mates_set_dout_all(h, mates_dio3_mk1_1, 0);
   mates_close(h);
   return ((success == 4) ? 0 : -1);
}
```

Examples: mates_test_32.c.

UOS_STATUS mates_lamp_test (
MATES_HANDLE *h*,
int *addr*)

Execute the lamp test. See CR register documentation for details.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.

Note This function blocks until entire lamp test sequence is executed. For MK1 nodes this lasts ca 5 seconds.

Example usage (mates_test_33.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   int success = 0;
   if (mates_discover_single_node(h, mates_diox_mk1_1) !=
      UOS_STATUS_OK)
   {
      mates_print_errors();
      mates_close(h);
      return (0);
   }

   /* Locate node by blinking its LEDs. */
   if (mates_lamp_test(h, mates_diox_mk1_1) ==
      UOS_STATUS_OK)
   {
      success++;
   }
   mates_close(h);
   return ((success == 1) ? 0 : -1);
}
```

Examples: mates_test_33.c.

UOS_STATUS mates_alcd_write (
MATES_HANDLE *h*,
int *addr*,
int *row*,
int *col*,
char * *text*)

Write text onto the alphanumeric LCD display.

Parameters

in	*h*	The MATES communication handle.
in	*addr*	The node CAN address.
in	*row*	The LCD row number (0 - based).
in	*col*	The LCD column number (0 - based).
in	*text*	The NUL - terminated text to write at the given position.

Returns Status of the operation.

Return values

UOS_STATUS_OK	when successful.
UOS_STATUS_ERANGE	when *row* or *col* are out of display's range.
UOS_STATUS_ETOOBIG	when text doesn't fit into the display.

Remarks This functionality is available only on MATES-UCC-MK1 devices.

Since mates.dll 2.4.0.0

Example usage (mates_test_40.c):

```
#include <stdio.h>
#include "mates.h"

int main(void)
{
   int success = 0;
   MATES_HANDLE h = mates_open("proxy.mon", 1);
   if (mates_discover_single_node(h, mates_ucc_mk1_1) ==
      UOS_STATUS_OK)
   {
      if (mates_alcd_write(h, mates_ucc_mk1_1, 0, 0, "Hello World!") ==
      UOS_STATUS_OK)
      {
         success++;
      }
      if (mates_alcd_write(h, mates_ucc_mk1_1, 1, 0, "MATES LCD") ==
      UOS_STATUS_OK)
      {
         success++;
      }
      mates_close(h);
      return (success == 2 ? 0 : -1);
   }
   else
   {
      return (0);
   }
}
```

Examples: mates_test_40.c.

char* mates_status_name (
UOS_STATUS *code*)

Translate MATES status code into string representation.

Parameters

in	*code*	The status code.

Returns The status name.

A.5 Mates.cs File Reference

C# wrapper for the MATES-DLL.

Data Structures

- class Mates

  MATES system wrapper class.

Namespaces

- namespace Viresco.Mates

  MATES system main classes.

Enumerations

A.5.1 Detailed Description

C# wrapper for the MATES-DLL.

A.6 viresco.mates Namespace Reference

Python wrapper for the MATES-DLL.

Namespaces

- regs

  MATES nodes registers definitions.

Data Structures

- class Mates

  The MATES-DLL Python wrapper class.
- class MatesConfigError

  Represents exception used by MATES to communicate configuration errors.
- class MatesException

  Represents exception used by MATES to communicate runtime errors.
- class MatesRangeError

  Represents exception used by MATES to signal range errors.
- class Node

  Represents abstract MATES node.
- class UccNode

  Represents single MATES-UCC-MK1 node.

A.6.1 Detailed Description

Python wrapper for the MATES-DLL.

```
Project:          MATES-DLL
Platform:         Python 2.7 / Python 3.x for Windows XP SP3, Windows 7,
                  Windows 8, Windows 10
```

A.7 MatesRegs.cs File Reference

MATES nodes registers definitions.

Namespaces

- namespace Viresco.Mates.Registers

  The MATES registers definitions.

Enumerations

A.7.1 Detailed Description

MATES nodes registers definitions.

A.8 viresco.mates.regs Namespace Reference

MATES nodes registers definitions.

Data Structures

- class MATES_REGS

  MATES registers numbers.

A.8.1 Detailed Description

MATES nodes registers definitions.

A.9 Mates Class Reference

The MATES-DLL Python wrapper class.

Public Member Functions

- def __init__ (self, filename=None, channel=None)

  Create the MATES instance.
- def __str__ (self)

  The string representation of the MATES object.
- def print_err (self)

  Print the accumulated error messages to the standard output.
- def get_err (self)

  Get the accumulated error description.
- def clear_errors (self)

  Clear the accumulated error messages.
- def start (self, filename, channel)

  Open the MATES communication channel.
- def stop (self)

  Finalize the MATES object.

- def open (self, filename, channel)

  Open the MATES communication channel.
- def open_buffer (self, conf_string, channel)

  Open the MATES communication channel.
- def close (self)

  Close the MATES communication interface.
- def dll_version (self)

  Get MATES DLL version.
- def dll_path (self)

  Get MATES DLL path.
- def monitor_version (self)

  Get the underlying uOS_monitor instance version.
- def monitor_path (self)

  Get the underlying uOS_monitor instance path.
- def version_str (self)

  Get MATES software version.
- def set_dout (self, addr, channel, value)

  Set value of an individual digital output.
- def set_dout_all (self, addr, value, mode=DIO_OUT_MODE_SET)

  Set value of all outputs.
- def get_dout (self, addr, channel)

  Get value of an individual digital output.
- def get_dout_all (self, addr)

  Get value of all digital outputs.
- def get_din (self, addr, channel)

  Get the value of an individual digital input.
- def get_din_all (self, addr)

  Get the value of all digital inputs.
- def toggle_dout (self, addr, channel)

  Toggle value of an individual digital output.
- def set_dac_raw (self, addr, channel, value)

  Set raw value of the DAC.
- def get_dac_raw (self, addr, channel)

  Get the raw value of the DAC.
- def set_dac (self, addr, channel, value)

  Set the value of the DAC.
- def get_dac (self, addr, channel)

  Get the value of the DAC.
- def get_adc (self, addr, channel)

  Get the value of the ADC (Volts).
- def get_adc_raw (self, addr, channel)

  Get the raw value of the ADC.

- def save_default (self, addr, channel)

  Save the current value of an output channel as a default value.
- def set_reg (self, addr, reg, val)

  Set regular (integral) MATES register.
- def set_regf (self, addr, reg, val)

  Set floating point MATES register.
- def get_reg (self, addr, reg)

  Get regular (integral) MATES register.
- def get_regf (self, addr, reg)

  Get floating point MATES register.
- def discover_node (self, addr)

  Check if the specified node is present on the bus.
- def node_info (self, addr)

  Get node information in form of a string.
- def enable_generators (self, addr, enable)

  Enable signal generators for a node.
- def setup_generator (self, addr, channel, period, rise_time, fall_time)

  Setup signal generator for a digital channel.
- def lamp_test (self, addr)

  Execute the lamp test.
- def alcd_write (self, addr, row, col, text)

  Write text onto the alphanumeric LCD display.
- def new_ucc_node (self, addr)

  Discover and create a new MATES-UCC-MK1 node.

A.9.1 Detailed Description

The MATES-DLL Python wrapper class.

A.9.2 Constructor & Destructor Documentation

def __**init**__ (
self,
filename = `None`,
channel = `None`)

Create the MATES instance.

If at least *filename* is specified the MATES communication channel will be opened immediately.

Parameters

`in`	*filename*	(str) - if specified defines the name of the monitor configuration file (∗.mon).
`in`	*channel*	(int) The communication channel number, use 0 for direct connections. Cannot be specified without specifying *filename*.

```
1 import mates
2
3 # Standard way of using:
4 m = mates.mates()
5 m.start("default.mon")
6 m.stop()
7
8 # Alternatively the with construct can be used:
9 with mates.mates("default.mon") as m:
10     print(m.get_err())
```

A.9.3 Member Function Documentation

def get_err (
self **)**

Get the accumulated error description.

Returns (str) The buffered errors or empty string.

def clear_errors (
self **)**

Clear the accumulated error messages.

Example (mates_test_05.py):

```
1 import time
2 import mates
3
4 with mates.Mates("proxy.mon", 1) as m:
5     # Wait for the device to be connected.
6     for i in range(10):
7         try:
8             if m.discover_single_node(mates.Mates_dio3_mk1_1) == True:
9                 print("Found")
10                 break
11         except:
12             # Clear the errors and continue.
13             m.clear_errors()
14         time.sleep(0.1)
```

def start (
self,
filename,
channel **)**

Open the MATES communication channel.

Parameters

in	*filename*	(str) - file name of the configuration file.
in	*channel*	(int) The associated channel number (as defined in configuration file).

Returns (bool) - True when successful, otherwise False.

This function can be omitted if MATES instance is created with at least file name (see constructor).

def open (
self,
filename,
channel **)**

Open the MATES communication channel.

Parameters

in	*filename*	(str) - file name of the configuration file.
in	*channel*	(int) The associated channel number (as defined in configuration file).

Returns (Mates) The current instance.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_01.py):

```
1 import mates
2 m = mates.Mates()
3
4 with m.open("proxy.mon", 1) as x:
5     x.print_err()
```

def open_buffer (
self,
conf_string,
channel **)**

Open the MATES communication channel.

Parameters

in	*conf_string*	(str) *uOS_monitor* XML configuration string.
in	*channel*	(int) The associated channel number.

Returns (Mates) The current instance.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_33.py):

```
1 import mates
2
3 cfg = r"""<?xml version="1.0" encoding="utf-8"?>
4 <uOS_monitor_configuration>
5   <local_address>1</local_address>
6   <named_pipe map="1">
7     <name>\\RAKIETA\pipe\MATES_PROXY_PIPE</name>
8     <timeout>5000</timeout>
9   </named_pipe>
10 </uOS_monitor_configuration>"""
11
12 m = mates.Mates()
13 m.open_buffer(cfg, 1)
14 print("Node 30 present: ", m.discover_node(30))
15
16 m.close()
```

def close (
self **)**

Close the MATES communication interface.

Example (mates_test_02.py):

```
1 import mates
2 m = mates.Mates("proxy.mon", 1)
3 m.close()
```

def dll_version (
self **)**

Get MATES DLL version.

Returns (int, int, int, int) The *mates.dll* file major version, minor version, revision and build number.

Example (mates_test_03.py):

```
1 import mates
2 m = mates.Mates()
3 version = m.monitor_version()
4 print("uOS_monitor version: {0}".format(".".join(map(str, version))))
```

def dll_path (
self **)**

Get MATES DLL path.

Returns str - the *mates.dll* file path.

Since mates.dll 2.3.0.0

Example (mates_test_29.py):

```
1 import mates
2 m = mates.Mates()
3 version = m.dll_version()
4 print("MATES dll version: {0}".format(".".join(map(str, version))))
```

def monitor_version (
self **)**

Get the underlying *uOS_monitor* instance version.

Returns (int, int, int, int) The *uOS_monitor* interface file major version, minor version, revision and build number.

Since mates.dll 2.0.1.0

Example (mates_test_30.py):

```
1 import mates
2 m = mates.Mates()
3 path = m.dll_path()
4 print("MATES dll path: {0}".format(path))
```

def monitor_path (
self **)**

Get the underlying *uOS_monitor* instance path.

Returns (str) The *uOS_monitor* interface path.

Since mates.dll 2.3.0.0

Example (mates_test_03.py):

```
1 import mates
2 m = mates.Mates()
3 version = m.monitor_version()
4 print("uOS_monitor version: {0}".format(".".join(map(str, version))))
```

def version_str (
self **)**

Get MATES software version.

Returns (str) The version string.

def set_dout (
self,
addr,
channel,
value **)**

Set value of an individual digital output.

Parameters

in	*addr*	(int) The CAN node address.
in	*channel*	(int) The DOUT channel number [0,19].
in	*value*	(int) The output value.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_04.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         m.set_dout(m.mates_dio3_mk1_1, 5, 1)
5         m.set_dout(m.mates_dio3_mk1_1, 5, 0)
6     elif m.discover_node(m.mates_diox_mk1_1):
7         m.set_dout(m.mates_diox_mk1_1, 5, 1)
8         m.set_dout(m.mates_diox_mk1_1, 5, 0)
```

def set_dout_all (
self,
addr,
value,
mode *= DIO_OUT_MODE_SET* **)**

Set value of all outputs.

Parameters

in	*addr*	(int) The CAN node address.
in	*value*	(int) The value to set.
in	*mode*	(int) The setting mode (see MATES manual): • 0: SET (OUT <- value) • 1: AND (OUT <- OUT & value) • 2: OR (OUT <- OUT | value) • 3: XOR (OUT <- OUT ^ value)

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_19.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         m.set_dout_all(m.mates_dio3_mk1_1, 0xAAAAA)
5         m.set_dout_all(m.mates_dio3_mk1_1, 0xFFFFF, mates.Mates.DIO_OUT_MODE_XOR)
6         print("Outputs (expecting 0x55555): 0x{0:X}".format(m.get_din_all(m.mates_dio3_mk1_1)))
```

def get_dout (
self,
addr,
***channel*)**

Get value of an individual digital output.

Parameters

in	*addr*	(int) The CAN node address.
in	*channel*	(int) The DOUT channel number [0,19].

Returns (int) The state of the output.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_06.py):

```
1 import random
2 import mates
3 with mates.Mates("proxy.mon", 1) as m:
4     if m.discover_node(m.mates_dio3_mk1_1):
5         m.set_dout(m.mates_dio3_mk1_1, 1, random.randint(0, 1))
6         print("Value set: {0}".format(m.get_dout(m.mates_dio3_mk1_1, 1)))
```

def get_dout_all (
self,
***addr*)**

Get value of all digital outputs.

Parameters

in	*addr*	(int) The CAN node address.

Returns (int) The state of all outputs, LSB is OUT01.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_21.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Set the outputs to some value.
5         m.set_dout_all(m.mates_dio3_mk1_1, 0xBAFED)
6         # Now read back.
7         print("Outputs: 0x{0:X}".format(m.get_dout_all(m.mates_dio3_mk1_1)))
```

def get_din (
self,
addr,
***channel*)**

Get the value of an individual digital input.

Parameters

in	*addr*	(int) The CAN node address.
in	*channel*	(int) The DOUT channel number [0,19].

Returns (int) The state of the input.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_07.py):

```
1 import time
2 import mates
3 with mates.Mates("proxy.mon", 1) as m:
4     if m.discover_node(m.mates_dio3_mk1_1):
5         # Wait for IN05 to become high.
6         for i in range(5):
7             if m.get_din(m.mates_dio3_mk1_1, 4) == 1:
8                 print("IN05 became high")
9                 break
10             time.sleep(0.1)
```

def get_din_all (
self,
***addr*)**

Get the value of all digital inputs.

Parameters

in	*addr*	(int) The CAN node address.

Returns (int) The state of the inputs, LSB is IN01.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_20.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Set the least significant nibble of the outputs to high state.
5         m.set_dout_all(m.mates_dio3_mk1_1, 0xF, mates.Mates.DIO_OUT_MODE_OR)
6         # See what the result is (assuming we have loopback installed).
7         print("Inputs: 0x{0:X}".format(m.get_din_all(m.mates_dio3_mk1_1)))
```

def toggle_dout (
self,
addr,
***channel*)**

Toggle value of an individual digital output.

Parameters

in	*addr*	(int) The CAN node address.

in	*channel*	(int) The DOUT channel number [0,19].

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_08.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(20):
4         for i in range(100):
5             # Toggle 20th channel of first DIO3 node.
6             m.toggle_dout(20, 19)
```

def set_dac_raw (
self,
addr,
channel,
value **)**

Set raw value of the DAC.

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) - DAC channel number [0,39].
in	*value*	(int) The raw DAC binary code to write.

Note This function ignores the DAC calibration data.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_09.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dac5_mk1_2):
4         # Set to ~0.5 V.
5         value = int((0.5 / 5.0) * 0x10000)
6         print("Setting raw DAC value: {0}".format(value))
7         m.set_dac_raw(m.mates_dac5_mk1_2, 39, value & 0xFFFF)
8
9         # When setting out of range exception is raised.
10         try:
11             m.set_dac_raw(m.mates_dac5_mk1_2, 39, 0xFFFF + 1)
12         except mates.MatesException as e:
13             print("Error setting DAC:")
14             print(e)
```

def get_dac_raw (
self,
addr,
channel **)**

Get the raw value of the DAC.

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) The DAC channel number [0,39].

Returns (int) The raw DAC value.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_10.py):

```
1 import random
2 import mates
3 with mates.Mates("proxy.mon", 1) as m:
4     if m.discover_node(m.mates_dac5_mk1_2):
5         # Set to random.
6         value = random.randint(0, 100)
7         m.set_dac_raw(m.mates_dac5_mk1_2, 39, value)
8         print("Raw DAC value: {0}".format(m.get_dac_raw(m.mates_dac5_mk1_2, 39)))
```

def set_dac (
self,
addr,
channel,
value **)**

Set the value of the DAC.

Takes calibration data into account.

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) The DAC channel number [0,39].
in	*value*	(float) The voltage value to set (Volts).

Note This function takes calibration data into account.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_11.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dac5_mk1_2):
4         # Set DAC voltage to 1 V if digital input is 0 or
5         # to 2 V if the digital input is 1.
6         if not m.get_din(m.mates_dio3_mk1_1, 0):
7             m.set_dac(m.mates_dac5_mk1_2, 0, 1.0)
8         else:
9             m.set_dac(m.mates_dac5_mk1_2, 0, 2.0)
```

def get_dac (
self,
addr,
channel **)**

Get the value of the DAC.

When using mates_set_dac() and mates_get_dac() in a pair, the verified value may be slightly different than the set value. This is due to the fact that the value is internally converted to single precision floating point value so precision of the double precision argument can be lost. This behaviour may change in the future.

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) DAC channel number [0,39].

Returns (float) - DAC voltage value in Volts.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_12.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dac5_mk1_2):
4         # Compare the value written and read.
5         value = 2.0 / 3.0
6         m.set_dac(m.mates_dac5_mk1_2, 10, value)
7         check = m.get_dac(m.mates_dac5_mk1_2, 10)
8         print("{0} vs {1}".format(value, check))
```

def get_adc (
self,
addr,
***channel*)**

Get the value of the ADC (Volts).

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) ADC channel number [0,39].

Returns (float) - ADC voltage value in Volts.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_23.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_adc5_mk1_1):
4         # Get voltage read from channel IN10.
5         value = m.get_adc(m.mates_adc5_mk1_1, 9)
6         print("Voltage at channel IN10: {0} V".format(value))
```

def get_adc_raw (
self,
addr,
***channel*)**

Get the raw value of the ADC.

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) - ADC channel number [0,39].

Returns (int) The raw ADC value.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_24.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_adc5_mk1_1):
4         # Get raw ADC code read from channel IN10.
5         value = m.get_adc_raw(m.mates_adc5_mk1_1, 9)
6         print("Raw value at channel IN10: {0}".format(value))
```

def save_default (
self,
addr,
***channel*)**

Save the current value of an output channel as a default value.

Parameters

in	*addr*	(int) The node address.
in	*channel*	(int) - output node channel number, [0,39] for DAC node, [0,19] for DIO3 or DIOX node.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_25.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Set channel OUT20 to high, save as default,
5         # then set to low and save as default.
6         m.set_dout(m.mates_dio3_mk1_1, 19, 1)
7         m.save_default(m.mates_dio3_mk1_1, 19)
8         m.set_dout(m.mates_dio3_mk1_1, 19, 0)
9         m.save_default(m.mates_dio3_mk1_1, 19)
```

def set_reg (
self,
addr,
reg,
***val*)**

Set regular (integral) MATES register.

Parameters

in	*addr*	(int) The node address.
in	*reg*	(int) The register number to access (see regs.MATES_REGS).
out	*val*	(int) The register value.

Exceptions

MatesException	Thrown when operation fails.
MatesRangeError	Thrown when written value was out of range.

Example (mates_test_13.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Set ODEF bit in CR register to set all outputs to their default values.
5         m.set_reg(m.mates_dio3_mk1_1, m.regs.REG_COMMON_CR, 1)
```

def set_regf (
self,
addr,
reg,
***val*)**

Set floating point MATES register.

Parameters

in	*addr*	(int) The node address.
in	*reg*	(int) The register number to access (see regs.MATES_REGS).
in	*val*	(float) The register value.

Exceptions

MatesException	Thrown when operation fails.
MatesRangeError	Thrown when written value was out of range.

Example (mates_test_17.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dac5_mk1_2):
4         # Set default values of all DAC channels to 0.1 V.
5         for channel in range(40):
6             m.set_regf(m.mates_dac5_mk1_2, m.regs.REG_MATES_DAC5_MK1_DEF01 + channel, 0.1)
```

def get_reg (
self,
addr,
***reg*)**

Get regular (integral) MATES register.

Parameters

in	*addr*	(int) The node address.
in	*reg*	(int) The register number to access (see regs.MATES_REGS).

Returns (int) the register value.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_15.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Get hardware revision for the device from the DIR register.
5         dir = m.get_reg(m.mates_dio3_mk1_1, m.regs.REG_COMMON_DIR)
6         version = (dir >> 24) & 0xFF
7         print("Hardware version: MK{0}".format(version))
```

def get_regf (
self,
addr,
***reg*)**

Get floating point MATES register.

Parameters

in	*addr*	(int) The node address.
in	*reg*	(int) The register number to access (see regs.MATES_REGS).

Returns (float) The register value.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_16.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dac5_mk1_2):
4         # Get default value of the first DAC output.
5         value = m.get_regf(m.mates_dac5_mk1_2, m.regs.REG_MATES_DAC5_MK1_DEF01)
6         print("Default voltage value: {0} V".format(value))
```

def discover_node (
self,
addr **)**

Check if the specified node is present on the bus.

Parameters

in	*addr*	(int) The CAN node address.

Returns (bool) True when device present, otherwise False.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_32.py):

```
1 import mates
2
3 with mates.Mates("proxy.mon", 1) as m:
4     for node in (20, 21, 40, 41):
5         if m.discover_node(node):
6             print("Discovered node #{0}".format(node))
7         else:
8             print("Node #{0} not found".format(node))
9
```

def node_info (
self,
addr **)**

Get node information in form of a string.

Parameters

in	*addr*	(int) The CAN node address.

Returns (str) The node information string.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_22.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     print("Connected nodes:")
4     for node in (20, 21, 50, 51):
5         try:
6             print(m.node_info(node))
7         except:
8             print("<not found>")
```

def enable_generators (
self,
addr,
enable **)**

Enable signal generators for a node.

Parameters

in	*addr*	(int) The CAN node address.
in	*enable*	(int) The generators enable state.

Remarks Only MATES-DIO3-MK1 support signal generators.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_26.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Disable the generators and then enable them,
5         # this resets all time bases of the generators.
6         m.enable_generators(m.mates_dio3_mk1_1, False)
7         m.enable_generators(m.mates_dio3_mk1_1, True)
8
```

def setup_generator (
self,
addr,
channel,
period,
rise_time,
fall_time **)**

Setup signal generator for a digital channel.

Parameters

in	*addr*	(int) The CAN node address.
in	*channel*	(int) The generator channel to setup [0,19].
in	*period*	(int) The waveform period in microseconds.
in	*rise_time*	(int) - signal rise time within the period in microseconds.
in	*fall_time*	(int) - signal fall time within the period in microseconds.

Note *period*, *rise_time* and *fall_time* have to be a multiple of 125 us.

Remarks Only MATES-DIO3-MK1 nodes support signal generators.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_27.py):

```
1 import time
2 import mates
3 with mates.Mates("proxy.mon", 1) as m:
4     if m.discover_node(m.mates_dio3_mk1_1):
5         # Generate quadrature waveform on OUT01 and OUT02 at 100 Hz.
6         m.enable_generators(m.mates_dio3_mk1_1, False)
7         m.setup_generator(m.mates_dio3_mk1_1, 0, 10000, 0, 5000)
8         m.setup_generator(m.mates_dio3_mk1_1, 1, 10000, 2500, 7500)
9         m.enable_generators(m.mates_dio3_mk1_1, True)
10         # Generate for 2 seconds.
11         time.sleep(2)
```

```
12          # Disable the generators and clear the outputs.
13          m.setup_generator(m.mates_dio3_mk1_1, 0, 0, 0, 0)
14          m.setup_generator(m.mates_dio3_mk1_1, 1, 0, 0, 0)
15          m.enable_generators(m.mates_dio3_mk1_1, False)
16          m.set_dout_all(m.mates_dio3_mk1_1, 0)
```

def lamp_test (
self,
addr **)**

Execute the lamp test.

See CR register documentation for details.

Parameters

in	*addr*	(int) The CAN node address.

Note This function blocks until entire lamp test sequence is executed. For MK1 nodes this lasts ca 5 seconds.

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_28.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_dio3_mk1_1):
4         # Locate node by blinking its LEDs.
5         m.lamp_test(m.mates_dio3_mk1_1)
```

def alcd_write (
self,
addr,
row,
col,
text **)**

Write text onto the alphanumeric LCD display.

Parameters

in	*addr*	(int) The node CAN address.
in	*row*	(int) The LCD row number (0 - based).
in	*col*	(int) The LCD column number (0 - based).
in	*text*	(str) Text to write at the given position.

Remarks This functionality is available only on MATES-UCC-MK1 devices.

Since mates.dll 2.4.0.0

Exceptions

MatesException	Thrown when operation fails.

Example (mates_test_31.py):

```
1 import mates
2 with mates.Mates("proxy.mon", 1) as m:
3     if m.discover_node(m.mates_ucc_mk1_1):
4         text = "Padded with spaces".ljust(32)
5         m.alcd_write(m.mates_ucc_mk1_1, 0, 0, text)
```

def new_ucc_node (
self,
***addr*)**

Discover and create a new MATES-UCC-MK1 node.

Parameters

`in`	*addr*	(int) The node address to use.

Returns (UccNode) The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

The documentation for this class was generated from the following file:

- __init__.py

A.10 MatesConfigError Class Reference

Represents exception used by MATES to communicate configuration errors.

A.10.1 Detailed Description

Represents exception used by MATES to communicate configuration errors.

The documentation for this class was generated from the following file:

- __init__.py

A.11 MatesException Class Reference

Represents exception used by MATES to communicate runtime errors.

A.11.1 Detailed Description

Represents exception used by MATES to communicate runtime errors.

The documentation for this class was generated from the following file:

- __init__.py

A.12 MatesRangeError Class Reference

Represents exception used by MATES to signal range errors.

A.12.1 Detailed Description

Represents exception used by MATES to signal range errors.

The documentation for this class was generated from the following file:

- __init__.py

A.13 Node Class Reference

Represents abstract MATES node.

A.13.1 Detailed Description

Represents abstract MATES node.

The documentation for this class was generated from the following file:

- __init__.py

A.14 UccNode Class Reference

Represents single MATES-UCC-MK1 node.

A.14.1 Detailed Description

Represents single MATES-UCC-MK1 node.

Remarks The UccNode constructor is not intended to be called directly. Use Mates.new_ucc_node instead.

The documentation for this class was generated from the following file:

- __init__.py

A.15 MATES_REGS Class Reference

MATES registers numbers.

Static Public Attributes

- int REG_COMMON_TEST = 0x0

  (0) TEST (#0x0, RWN, init.
- int REG_COMMON_DIR = 0x1

  (1) DIR (#0x1, RO, init.
- int REG_COMMON_BIR = 0x2

  (2) BIR (#0x2, RO, init.
- int REG_COMMON_FIR = 0x3

  (3) FIR (#0x3, RO, init.
- int REG_COMMON_SR = 0x4

  (4) SR (#0x4, RO, init.
- int REG_COMMON_CR = 0x5

  (5) CR (#0x5, RW, init.
- int REG_COMMON_TUT = 0x6

  (6) TUT (#0x6, RO, init.
- int REG_COMMON_CUT = 0x7

  (7) CUT (#0x7, RO, init.
- int REG_COMMON_FUT = 0x8

  (8) FUT (#0x8, RO, init.
- int REG_COMMON_DPD = 0x9

  (9) DPD (#0x9, RO, init.
- int REG_COMMON_PUC = 0xA

  (10) PUC (#0xA, RO, init.
- int REG_COMMON_HIR = 0xB

(11) HIR (#0xB, RO, init.

- int REG_COMMON_CAPS01 = 0xC

(12) CAPS01 (#0xC, RO, init.

- int REG_COMMON_CAPS02 = 0xD

(13) CAPS02 (#0xD, RO, init.

- int REG_MATES_DIO3_MK1_ODEF = 0x200

(512) ODEF (#0x200, RWN, init.

- int REG_MATES_DIO3_MK1_ODIS = 0x201

(513) ODIS (#0x201, RW, init.

- int REG_MATES_DIO3_MK1_PRD01 = 0x202

(514) PRD01 (#0x202, RW, init.

- int REG_MATES_DIO3_MK1_PRD02 = 0x203

(515) PRD02 (#0x203, RW, init.

- int REG_MATES_DIO3_MK1_PRD03 = 0x204

(516) PRD03 (#0x204, RW, init.

- int REG_MATES_DIO3_MK1_PRD04 = 0x205

(517) PRD04 (#0x205, RW, init.

- int REG_MATES_DIO3_MK1_PRD05 = 0x206

(518) PRD05 (#0x206, RW, init.

- int REG_MATES_DIO3_MK1_PRD06 = 0x207

(519) PRD06 (#0x207, RW, init.

- int REG_MATES_DIO3_MK1_PRD07 = 0x208

(520) PRD07 (#0x208, RW, init.

- int REG_MATES_DIO3_MK1_PRD08 = 0x209

(521) PRD08 (#0x209, RW, init.

- int REG_MATES_DIO3_MK1_PRD09 = 0x20A

(522) PRD09 (#0x20A, RW, init.

- int REG_MATES_DIO3_MK1_PRD10 = 0x20B

(523) PRD10 (#0x20B, RW, init.

- int REG_MATES_DIO3_MK1_PRD11 = 0x20C

(524) PRD11 (#0x20C, RW, init.

- int REG_MATES_DIO3_MK1_PRD12 = 0x20D

(525) PRD12 (#0x20D, RW, init.

- int REG_MATES_DIO3_MK1_PRD13 = 0x20E

(526) PRD13 (#0x20E, RW, init.

- int REG_MATES_DIO3_MK1_PRD14 = 0x20F

(527) PRD14 (#0x20F, RW, init.

- int REG_MATES_DIO3_MK1_PRD15 = 0x210

(528) PRD15 (#0x210, RW, init.

- int REG_MATES_DIO3_MK1_PRD16 = 0x211

(529) PRD16 (#0x211, RW, init.

- int REG_MATES_DIO3_MK1_PRD17 = 0x212

(530) PRD17 (#0x212, RW, init.

- int REG_MATES_DIO3_MK1_PRD18 = 0x213

(531) PRD18 (#0x213, RW, init.

- int REG_MATES_DIO3_MK1_PRD19 = 0x214

(532) PRD19 (#0x214, RW, init.

- int REG_MATES_DIO3_MK1_PRD20 = 0x215

(533) PRD20 (#0x215, RW, init.

- int REG_MATES_DIO3_MK1_TRISE01 = 0x216

(534) TRISE01 (#0x216, RW, init.

- int REG_MATES_DIO3_MK1_TRISE02 = 0x217

(535) TRISE02 (#0x217, RW, init.

- int REG_MATES_DIO3_MK1_TRISE03 = 0x218

(536) TRISE03 (#0x218, RW, init.

- int REG_MATES_DIO3_MK1_TRISE04 = 0x219

(537) TRISE04 (#0x219, RW, init.

- int REG_MATES_DIO3_MK1_TRISE05 = 0x21A

(538) TRISE05 (#0x21A, RW, init.

- int REG_MATES_DIO3_MK1_TRISE06 = 0x21B

(539) TRISE06 (#0x21B, RW, init.

- int REG_MATES_DIO3_MK1_TRISE07 = 0x21C

(540) TRISE07 (#0x21C, RW, init.

- int REG_MATES_DIO3_MK1_TRISE08 = 0x21D

(541) TRISE08 (#0x21D, RW, init.

- int REG_MATES_DIO3_MK1_TRISE09 = 0x21E

(542) TRISE09 (#0x21E, RW, init.

- int REG_MATES_DIO3_MK1_TRISE10 = 0x21F

(543) TRISE10 (#0x21F, RW, init.

- int REG_MATES_DIO3_MK1_TRISE11 = 0x220

(544) TRISE11 (#0x220, RW, init.

- int REG_MATES_DIO3_MK1_TRISE12 = 0x221

(545) TRISE12 (#0x221, RW, init.

- int REG_MATES_DIO3_MK1_TRISE13 = 0x222

(546) TRISE13 (#0x222, RW, init.

- int REG_MATES_DIO3_MK1_TRISE14 = 0x223

(547) TRISE14 (#0x223, RW, init.

- int REG_MATES_DIO3_MK1_TRISE15 = 0x224

(548) TRISE15 (#0x224, RW, init.

- int REG_MATES_DIO3_MK1_TRISE16 = 0x225

(549) TRISE16 (#0x225, RW, init.

- int REG_MATES_DIO3_MK1_TRISE17 = 0x226

(550) TRISE17 (#0x226, RW, init.

- int REG_MATES_DIO3_MK1_TRISE18 = 0x227

(551) TRISE18 (#0x227, RW, init.

- int REG_MATES_DIO3_MK1_TRISE19 = 0x228

(552) TRISE19 (#0x228, RW, init.

- int REG_MATES_DIO3_MK1_TRISE20 = 0x229

(553) TRISE20 (#0x229, RW, init.

- int REG_MATES_DIO3_MK1_TFALL01 = 0x22A

(554) TFALL01 (#0x22A, RW, init.

- int REG_MATES_DIO3_MK1_TFALL02 = 0x22B

(555) TFALL02 (#0x22B, RW, init.

- int REG_MATES_DIO3_MK1_TFALL03 = 0x22C

(556) TFALL03 (#0x22C, RW, init.

- int REG_MATES_DIO3_MK1_TFALL04 = 0x22D

(557) TFALL04 (#0x22D, RW, init.

- int REG_MATES_DIO3_MK1_TFALL05 = 0x22E

(558) TFALL05 (#0x22E, RW, init.

- int REG_MATES_DIO3_MK1_TFALL06 = 0x22F

(559) TFALL06 (#0x22F, RW, init.

- int REG_MATES_DIO3_MK1_TFALL07 = 0x230

(560) TFALL07 (#0x230, RW, init.

- int REG_MATES_DIO3_MK1_TFALL08 = 0x231

(561) TFALL08 (#0x231, RW, init.

- int REG_MATES_DIO3_MK1_TFALL09 = 0x232

(562) TFALL09 (#0x232, RW, init.

- int REG_MATES_DIO3_MK1_TFALL10 = 0x233

(563) TFALL10 (#0x233, RW, init.

- int REG_MATES_DIO3_MK1_TFALL11 = 0x234

(564) TFALL11 (#0x234, RW, init.

- int REG_MATES_DIO3_MK1_TFALL12 = 0x235

(565) TFALL12 (#0x235, RW, init.

- int REG_MATES_DIO3_MK1_TFALL13 = 0x236

(566) TFALL13 (#0x236, RW, init.

- int REG_MATES_DIO3_MK1_TFALL14 = 0x237

(567) TFALL14 (#0x237, RW, init.

- int REG_MATES_DIO3_MK1_TFALL15 = 0x238

(568) TFALL15 (#0x238, RW, init.

- int REG_MATES_DIO3_MK1_TFALL16 = 0x239

(569) TFALL16 (#0x239, RW, init.

- int REG_MATES_DIO3_MK1_TFALL17 = 0x23A

(570) TFALL17 (#0x23A, RW, init.

- int REG_MATES_DIO3_MK1_TFALL18 = 0x23B

(571) TFALL18 (#0x23B, RW, init.

- int REG_MATES_DIO3_MK1_TFALL19 = 0x23C

(572) TFALL19 (#0x23C, RW, init.

- int REG_MATES_DIO3_MK1_TFALL20 = 0x23D

(573) TFALL20 (#0x23D, RW, init.

- int REG_MATES_DIO3_MK1_ORP = 0x23E

(574) ORP (#0x23E, RWN, init.

- int REG_MATES_DIO3_MK1_ORRT = 0x23F

(575) ORRT (#0x23F, RO, init.

- int REG_MATES_DIO3_MK1_PSEL = 0x240

(576) PSEL (#0x240, RWN, init.

- int REG_MATES_DIO3_MK1_CANID = 0x241

(577) CANID (#0x241, RWN, init.

- int REG_MATES_DAC5_MK1_DEF01 = 0x500

(1280) DEF01 (#0x500, RWN, init.

- int REG_MATES_DAC5_MK1_DEF02 = 0x501

(1281) DEF02 (#0x501, RWN, init.

- int REG_MATES_DAC5_MK1_DEF03 = 0x502

(1282) DEF03 (#0x502, RWN, init.

- int REG_MATES_DAC5_MK1_DEF04 = 0x503

(1283) DEF04 (#0x503, RWN, init.

- int REG_MATES_DAC5_MK1_DEF05 = 0x504

(1284) DEF05 (#0x504, RWN, init.

- int REG_MATES_DAC5_MK1_DEF06 = 0x505

(1285) DEF06 (#0x505, RWN, init.

- int REG_MATES_DAC5_MK1_DEF07 = 0x506

(1286) DEF07 (#0x506, RWN, init.

- int REG_MATES_DAC5_MK1_DEF08 = 0x507

(1287) DEF08 (#0x507, RWN, init.

- int REG_MATES_DAC5_MK1_DEF09 = 0x508

(1288) DEF09 (#0x508, RWN, init.

- int REG_MATES_DAC5_MK1_DEF10 = 0x509

(1289) DEF10 (#0x509, RWN, init.

- int REG_MATES_DAC5_MK1_DEF11 = 0x50A

(1290) DEF11 (#0x50A, RWN, init.

- int REG_MATES_DAC5_MK1_DEF12 = 0x50B

(1291) DEF12 (#0x50B, RWN, init.

- int REG_MATES_DAC5_MK1_DEF13 = 0x50C

(1292) DEF13 (#0x50C, RWN, init.

- int REG_MATES_DAC5_MK1_DEF14 = 0x50D

(1293) DEF14 (#0x50D, RWN, init.

- int REG_MATES_DAC5_MK1_DEF15 = 0x50E

(1294) DEF15 (#0x50E, RWN, init.

- int REG_MATES_DAC5_MK1_DEF16 = 0x50F

(1295) DEF16 (#0x50F, RWN, init.

- int REG_MATES_DAC5_MK1_DEF17 = 0x510

(1296) DEF17 (#0x510, RWN, init.

- int REG_MATES_DAC5_MK1_DEF18 = 0x511

(1297) DEF18 (#0x511, RWN, init.

- int REG_MATES_DAC5_MK1_DEF19 = 0x512

(1298) DEF19 (#0x512, RWN, init.

- int REG_MATES_DAC5_MK1_DEF20 = 0x513

(1299) DEF20 (#0x513, RWN, init.

- int REG_MATES_DAC5_MK1_DEF21 = 0x514

(1300) DEF21 (#0x514, RWN, init.

- int REG_MATES_DAC5_MK1_DEF22 = 0x515

(1301) DEF22 (#0x515, RWN, init.

- int REG_MATES_DAC5_MK1_DEF23 = 0x516

(1302) DEF23 (#0x516, RWN, init.

- int REG_MATES_DAC5_MK1_DEF24 = 0x517

(1303) DEF24 (#0x517, RWN, init.

- int REG_MATES_DAC5_MK1_DEF25 = 0x518

(1304) DEF25 (#0x518, RWN, init.

- int REG_MATES_DAC5_MK1_DEF26 = 0x519

(1305) DEF26 (#0x519, RWN, init.

- int REG_MATES_DAC5_MK1_DEF27 = 0x51A

(1306) DEF27 (#0x51A, RWN, init.

- int REG_MATES_DAC5_MK1_DEF28 = 0x51B

(1307) DEF28 (#0x51B, RWN, init.

- int REG_MATES_DAC5_MK1_DEF29 = 0x51C

(1308) DEF29 (#0x51C, RWN, init.

- int REG_MATES_DAC5_MK1_DEF30 = 0x51D

(1309) DEF30 (#0x51D, RWN, init.

- int REG_MATES_DAC5_MK1_DEF31 = 0x51E

(1310) DEF31 (#0x51E, RWN, init.

- int REG_MATES_DAC5_MK1_DEF32 = 0x51F

(1311) DEF32 (#0x51F, RWN, init.

- int REG_MATES_DAC5_MK1_DEF33 = 0x520

(1312) DEF33 (#0x520, RWN, init.

- int REG_MATES_DAC5_MK1_DEF34 = 0x521

(1313) DEF34 (#0x521, RWN, init.

- int REG_MATES_DAC5_MK1_DEF35 = 0x522

(1314) DEF35 (#0x522, RWN, init.

- int REG_MATES_DAC5_MK1_DEF36 = 0x523

(1315) DEF36 (#0x523, RWN, init.

- int REG_MATES_DAC5_MK1_DEF37 = 0x524

(1316) DEF37 (#0x524, RWN, init.

- int REG_MATES_DAC5_MK1_DEF38 = 0x525

(1317) DEF38 (#0x525, RWN, init.

- int REG_MATES_DAC5_MK1_DEF39 = 0x526

(1318) DEF39 (#0x526, RWN, init.

- int REG_MATES_DAC5_MK1_DEF40 = 0x527

(1319) DEF40 (#0x527, RWN, init.

- int REG_MATES_DAC5_MK1_MIN01 = 0x528

(1320) MIN01 (#0x528, RWN, init.

- int REG_MATES_DAC5_MK1_MIN02 = 0x529

(1321) MIN02 (#0x529, RWN, init.

- int REG_MATES_DAC5_MK1_MIN03 = 0x52A

(1322) MIN03 (#0x52A, RWN, init.

- int REG_MATES_DAC5_MK1_MIN04 = 0x52B

(1323) MIN04 (#0x52B, RWN, init.

- int REG_MATES_DAC5_MK1_MIN05 = 0x52C

(1324) MIN05 (#0x52C, RWN, init.

- int REG_MATES_DAC5_MK1_MIN06 = 0x52D

(1325) MIN06 (#0x52D, RWN, init.

- int REG_MATES_DAC5_MK1_MIN07 = 0x52E

(1326) MIN07 (#0x52E, RWN, init.

- int REG_MATES_DAC5_MK1_MIN08 = 0x52F

(1327) MIN08 (#0x52F, RWN, init.

- int REG_MATES_DAC5_MK1_MIN09 = 0x530

(1328) MIN09 (#0x530, RWN, init.

- int REG_MATES_DAC5_MK1_MIN10 = 0x531

(1329) MIN10 (#0x531, RWN, init.

- int REG_MATES_DAC5_MK1_MIN11 = 0x532

(1330) MIN11 (#0x532, RWN, init.

- int REG_MATES_DAC5_MK1_MIN12 = 0x533

(1331) MIN12 (#0x533, RWN, init.

- int REG_MATES_DAC5_MK1_MIN13 = 0x534

(1332) MIN13 (#0x534, RWN, init.

- int REG_MATES_DAC5_MK1_MIN14 = 0x535

(1333) MIN14 (#0x535, RWN, init.

- int REG_MATES_DAC5_MK1_MIN15 = 0x536

(1334) MIN15 (#0x536, RWN, init.

- int REG_MATES_DAC5_MK1_MIN16 = 0x537

(1335) MIN16 (#0x537, RWN, init.

- int REG_MATES_DAC5_MK1_MIN17 = 0x538

(1336) MIN17 (#0x538, RWN, init.

- int REG_MATES_DAC5_MK1_MIN18 = 0x539

(1337) MIN18 (#0x539, RWN, init.

- int REG_MATES_DAC5_MK1_MIN19 = 0x53A

(1338) MIN19 (#0x53A, RWN, init.

- int REG_MATES_DAC5_MK1_MIN20 = 0x53B

(1339) MIN20 (#0x53B, RWN, init.

- int REG_MATES_DAC5_MK1_MIN21 = 0x53C

(1340) MIN21 (#0x53C, RWN, init.

- int REG_MATES_DAC5_MK1_MIN22 = 0x53D

(1341) MIN22 (#0x53D, RWN, init.

- int REG_MATES_DAC5_MK1_MIN23 = 0x53E

(1342) MIN23 (#0x53E, RWN, init.

- int REG_MATES_DAC5_MK1_MIN24 = 0x53F

(1343) MIN24 (#0x53F, RWN, init.

- int REG_MATES_DAC5_MK1_MIN25 = 0x540

(1344) MIN25 (#0x540, RWN, init.

- int REG_MATES_DAC5_MK1_MIN26 = 0x541

(1345) MIN26 (#0x541, RWN, init.

- int REG_MATES_DAC5_MK1_MIN27 = 0x542

(1346) MIN27 (#0x542, RWN, init.

- int REG_MATES_DAC5_MK1_MIN28 = 0x543

(1347) MIN28 (#0x543, RWN, init.

- int REG_MATES_DAC5_MK1_MIN29 = 0x544

(1348) MIN29 (#0x544, RWN, init.

- int REG_MATES_DAC5_MK1_MIN30 = 0x545

(1349) MIN30 (#0x545, RWN, init.

- int REG_MATES_DAC5_MK1_MIN31 = 0x546

(1350) MIN31 (#0x546, RWN, init.

- int REG_MATES_DAC5_MK1_MIN32 = 0x547

(1351) MIN32 (#0x547, RWN, init.

- int REG_MATES_DAC5_MK1_MIN33 = 0x548

(1352) MIN33 (#0x548, RWN, init.

- int REG_MATES_DAC5_MK1_MIN34 = 0x549

(1353) MIN34 (#0x549, RWN, init.

- int REG_MATES_DAC5_MK1_MIN35 = 0x54A

(1354) MIN35 (#0x54A, RWN, init.

- int REG_MATES_DAC5_MK1_MIN36 = 0x54B

(1355) MIN36 (#0x54B, RWN, init.

- int REG_MATES_DAC5_MK1_MIN37 = 0x54C

(1356) MIN37 (#0x54C, RWN, init.

- int REG_MATES_DAC5_MK1_MIN38 = 0x54D

(1357) MIN38 (#0x54D, RWN, init.

- int REG_MATES_DAC5_MK1_MIN39 = 0x54E

(1358) MIN39 (#0x54E, RWN, init.

- int REG_MATES_DAC5_MK1_MIN40 = 0x54F

(1359) MIN40 (#0x54F, RWN, init.

- int REG_MATES_DAC5_MK1_MAX01 = 0x550

(1360) MAX01 (#0x550, RWN, init.

- int REG_MATES_DAC5_MK1_MAX02 = 0x551

(1361) MAX02 (#0x551, RWN, init.

- int REG_MATES_DAC5_MK1_MAX03 = 0x552

(1362) MAX03 (#0x552, RWN, init.

- int REG_MATES_DAC5_MK1_MAX04 = 0x553

(1363) MAX04 (#0x553, RWN, init.

- int REG_MATES_DAC5_MK1_MAX05 = 0x554

(1364) MAX05 (#0x554, RWN, init.

- int REG_MATES_DAC5_MK1_MAX06 = 0x555

(1365) MAX06 (#0x555, RWN, init.

- int REG_MATES_DAC5_MK1_MAX07 = 0x556

(1366) MAX07 (#0x556, RWN, init.

- int REG_MATES_DAC5_MK1_MAX08 = 0x557

(1367) MAX08 (#0x557, RWN, init.

- int REG_MATES_DAC5_MK1_MAX09 = 0x558

(1368) MAX09 (#0x558, RWN, init.

- int REG_MATES_DAC5_MK1_MAX10 = 0x559

(1369) MAX10 (#0x559, RWN, init.

- int REG_MATES_DAC5_MK1_MAX11 = 0x55A

(1370) MAX11 (#0x55A, RWN, init.

- int REG_MATES_DAC5_MK1_MAX12 = 0x55B

(1371) MAX12 (#0x55B, RWN, init.

- int REG_MATES_DAC5_MK1_MAX13 = 0x55C

(1372) MAX13 (#0x55C, RWN, init.

- int REG_MATES_DAC5_MK1_MAX14 = 0x55D

(1373) MAX14 (#0x55D, RWN, init.

- int REG_MATES_DAC5_MK1_MAX15 = 0x55E

(1374) MAX15 (#0x55E, RWN, init.

- int REG_MATES_DAC5_MK1_MAX16 = 0x55F

(1375) MAX16 (#0x55F, RWN, init.

- int REG_MATES_DAC5_MK1_MAX17 = 0x560

(1376) MAX17 (#0x560, RWN, init.

- int REG_MATES_DAC5_MK1_MAX18 = 0x561

(1377) MAX18 (#0x561, RWN, init.

- int REG_MATES_DAC5_MK1_MAX19 = 0x562

(1378) MAX19 (#0x562, RWN, init.

- int REG_MATES_DAC5_MK1_MAX20 = 0x563

(1379) MAX20 (#0x563, RWN, init.

- int REG_MATES_DAC5_MK1_MAX21 = 0x564

(1380) MAX21 (#0x564, RWN, init.

- int REG_MATES_DAC5_MK1_MAX22 = 0x565

(1381) MAX22 (#0x565, RWN, init.

- int REG_MATES_DAC5_MK1_MAX23 = 0x566

(1382) MAX23 (#0x566, RWN, init.

- int REG_MATES_DAC5_MK1_MAX24 = 0x567

(1383) MAX24 (#0x567, RWN, init.

- int REG_MATES_DAC5_MK1_MAX25 = 0x568

(1384) MAX25 (#0x568, RWN, init.

- int REG_MATES_DAC5_MK1_MAX26 = 0x569

(1385) MAX26 (#0x569, RWN, init.

- int REG_MATES_DAC5_MK1_MAX27 = 0x56A

(1386) MAX27 (#0x56A, RWN, init.

- int REG_MATES_DAC5_MK1_MAX28 = 0x56B

(1387) MAX28 (#0x56B, RWN, init.

- int REG_MATES_DAC5_MK1_MAX29 = 0x56C

(1388) MAX29 (#0x56C, RWN, init.

- int REG_MATES_DAC5_MK1_MAX30 = 0x56D

(1389) MAX30 (#0x56D, RWN, init.

- int REG_MATES_DAC5_MK1_MAX31 = 0x56E

(1390) MAX31 (#0x56E, RWN, init.

- int REG_MATES_DAC5_MK1_MAX32 = 0x56F

(1391) MAX32 (#0x56F, RWN, init.

- int REG_MATES_DAC5_MK1_MAX33 = 0x570

(1392) MAX33 (#0x570, RWN, init.

- int REG_MATES_DAC5_MK1_MAX34 = 0x571

(1393) MAX34 (#0x571, RWN, init.

- int REG_MATES_DAC5_MK1_MAX35 = 0x572

(1394) MAX35 (#0x572, RWN, init.

- int REG_MATES_DAC5_MK1_MAX36 = 0x573

(1395) MAX36 (#0x573, RWN, init.

- int REG_MATES_DAC5_MK1_MAX37 = 0x574

(1396) MAX37 (#0x574, RWN, init.

- int REG_MATES_DAC5_MK1_MAX38 = 0x575

(1397) MAX38 (#0x575, RWN, init.

- int REG_MATES_DAC5_MK1_MAX39 = 0x576

(1398) MAX39 (#0x576, RWN, init.

- int REG_MATES_DAC5_MK1_MAX40 = 0x577

(1399) MAX40 (#0x577, RWN, init.

- int REG_MATES_DAC5_MK1_COF01 = 0x578

(1400) COF01 (#0x578, RWN, init.

- int REG_MATES_DAC5_MK1_COF02 = 0x579

(1401) COF02 (#0x579, RWN, init.

- int REG_MATES_DAC5_MK1_COF03 = 0x57A

(1402) COF03 (#0x57A, RWN, init.

- int REG_MATES_DAC5_MK1_COF04 = 0x57B

(1403) COF04 (#0x57B, RWN, init.

- int REG_MATES_DAC5_MK1_COF05 = 0x57C

(1404) COF05 (#0x57C, RWN, init.

- int REG_MATES_DAC5_MK1_COF06 = 0x57D

(1405) COF06 (#0x57D, RWN, init.

- int REG_MATES_DAC5_MK1_COF07 = 0x57E

(1406) COF07 (#0x57E, RWN, init.

- int REG_MATES_DAC5_MK1_COF08 = 0x57F

(1407) COF08 (#0x57F, RWN, init.

- int REG_MATES_DAC5_MK1_COF09 = 0x580

(1408) COF09 (#0x580, RWN, init.

- int REG_MATES_DAC5_MK1_COF10 = 0x581

(1409) COF10 (#0x581, RWN, init.

- int REG_MATES_DAC5_MK1_COF11 = 0x582

(1410) COF11 (#0x582, RWN, init.

- int REG_MATES_DAC5_MK1_COF12 = 0x583

(1411) COF12 (#0x583, RWN, init.

- int REG_MATES_DAC5_MK1_COF13 = 0x584

(1412) COF13 (#0x584, RWN, init.

- int REG_MATES_DAC5_MK1_COF14 = 0x585

(1413) COF14 (#0x585, RWN, init.

- int REG_MATES_DAC5_MK1_COF15 = 0x586

(1414) COF15 (#0x586, RWN, init.

- int REG_MATES_DAC5_MK1_COF16 = 0x587

(1415) COF16 (#0x587, RWN, init.

- int REG_MATES_DAC5_MK1_COF17 = 0x588

(1416) COF17 (#0x588, RWN, init.

- int REG_MATES_DAC5_MK1_COF18 = 0x589

(1417) COF18 (#0x589, RWN, init.

- int REG_MATES_DAC5_MK1_COF19 = 0x58A

(1418) COF19 (#0x58A, RWN, init.

- int REG_MATES_DAC5_MK1_COF20 = 0x58B

(1419) COF20 (#0x58B, RWN, init.

- int REG_MATES_DAC5_MK1_COF21 = 0x58C

(1420) COF21 (#0x58C, RWN, init.

- int REG_MATES_DAC5_MK1_COF22 = 0x58D

(1421) COF22 (#0x58D, RWN, init.

- int REG_MATES_DAC5_MK1_COF23 = 0x58E

(1422) COF23 (#0x58E, RWN, init.

- int REG_MATES_DAC5_MK1_COF24 = 0x58F

(1423) COF24 (#0x58F, RWN, init.

- int REG_MATES_DAC5_MK1_COF25 = 0x590

(1424) COF25 (#0x590, RWN, init.

- int REG_MATES_DAC5_MK1_COF26 = 0x591

(1425) COF26 (#0x591, RWN, init.

- int REG_MATES_DAC5_MK1_COF27 = 0x592

(1426) COF27 (#0x592, RWN, init.

- int REG_MATES_DAC5_MK1_COF28 = 0x593

(1427) COF28 (#0x593, RWN, init.

- int REG_MATES_DAC5_MK1_COF29 = 0x594

(1428) COF29 (#0x594, RWN, init.

- int REG_MATES_DAC5_MK1_COF30 = 0x595

(1429) COF30 (#0x595, RWN, init.

- int REG_MATES_DAC5_MK1_COF31 = 0x596

(1430) COF31 (#0x596, RWN, init.

- int REG_MATES_DAC5_MK1_COF32 = 0x597

(1431) COF32 (#0x597, RWN, init.

- int REG_MATES_DAC5_MK1_COF33 = 0x598

(1432) COF33 (#0x598, RWN, init.

- int REG_MATES_DAC5_MK1_COF34 = 0x599

(1433) COF34 (#0x599, RWN, init.

- int REG_MATES_DAC5_MK1_COF35 = 0x59A

(1434) COF35 (#0x59A, RWN, init.

- int REG_MATES_DAC5_MK1_COF36 = 0x59B

(1435) COF36 (#0x59B, RWN, init.

- int REG_MATES_DAC5_MK1_COF37 = 0x59C

(1436) COF37 (#0x59C, RWN, init.

- int REG_MATES_DAC5_MK1_COF38 = 0x59D

(1437) COF38 (#0x59D, RWN, init.

- int REG_MATES_DAC5_MK1_COF39 = 0x59E

(1438) COF39 (#0x59E, RWN, init.

- int REG_MATES_DAC5_MK1_COF40 = 0x59F

(1439) COF40 (#0x59F, RWN, init.

- int REG_MATES_DAC5_MK1_CGA01 = 0x5A0

(1440) CGA01 (#0x5A0, RWN, init.

- int REG_MATES_DAC5_MK1_CGA02 = 0x5A1

(1441) CGA02 (#0x5A1, RWN, init.

- int REG_MATES_DAC5_MK1_CGA03 = 0x5A2

(1442) CGA03 (#0x5A2, RWN, init.

- int REG_MATES_DAC5_MK1_CGA04 = 0x5A3

(1443) CGA04 (#0x5A3, RWN, init.

- int REG_MATES_DAC5_MK1_CGA05 = 0x5A4

(1444) CGA05 (#0x5A4, RWN, init.

- int REG_MATES_DAC5_MK1_CGA06 = 0x5A5

(1445) CGA06 (#0x5A5, RWN, init.

- int REG_MATES_DAC5_MK1_CGA07 = 0x5A6

(1446) CGA07 (#0x5A6, RWN, init.

- int REG_MATES_DAC5_MK1_CGA08 = 0x5A7

(1447) CGA08 (#0x5A7, RWN, init.

- int REG_MATES_DAC5_MK1_CGA09 = 0x5A8

(1448) CGA09 (#0x5A8, RWN, init.

- int REG_MATES_DAC5_MK1_CGA10 = 0x5A9

(1449) CGA10 (#0x5A9, RWN, init.

- int REG_MATES_DAC5_MK1_CGA11 = 0x5AA

(1450) CGA11 (#0x5AA, RWN, init.

- int REG_MATES_DAC5_MK1_CGA12 = 0x5AB

(1451) CGA12 (#0x5AB, RWN, init.

- int REG_MATES_DAC5_MK1_CGA13 = 0x5AC

(1452) CGA13 (#0x5AC, RWN, init.

- int REG_MATES_DAC5_MK1_CGA14 = 0x5AD

(1453) CGA14 (#0x5AD, RWN, init.

- int REG_MATES_DAC5_MK1_CGA15 = 0x5AE

(1454) CGA15 (#0x5AE, RWN, init.

- int REG_MATES_DAC5_MK1_CGA16 = 0x5AF

(1455) CGA16 (#0x5AF, RWN, init.

- int REG_MATES_DAC5_MK1_CGA17 = 0x5B0

(1456) CGA17 (#0x5B0, RWN, init.

- int REG_MATES_DAC5_MK1_CGA18 = 0x5B1

(1457) CGA18 (#0x5B1, RWN, init.

- int REG_MATES_DAC5_MK1_CGA19 = 0x5B2

(1458) CGA19 (#0x5B2, RWN, init.

- int REG_MATES_DAC5_MK1_CGA20 = 0x5B3

(1459) CGA20 (#0x5B3, RWN, init.

- int REG_MATES_DAC5_MK1_CGA21 = 0x5B4

(1460) CGA21 (#0x5B4, RWN, init.

- int REG_MATES_DAC5_MK1_CGA22 = 0x5B5

(1461) CGA22 (#0x5B5, RWN, init.

- int REG_MATES_DAC5_MK1_CGA23 = 0x5B6

(1462) CGA23 (#0x5B6, RWN, init.

- int REG_MATES_DAC5_MK1_CGA24 = 0x5B7

(1463) CGA24 (#0x5B7, RWN, init.

- int REG_MATES_DAC5_MK1_CGA25 = 0x5B8

(1464) CGA25 (#0x5B8, RWN, init.

- int REG_MATES_DAC5_MK1_CGA26 = 0x5B9

(1465) CGA26 (#0x5B9, RWN, init.

- int REG_MATES_DAC5_MK1_CGA27 = 0x5BA

(1466) CGA27 (#0x5BA, RWN, init.

- int REG_MATES_DAC5_MK1_CGA28 = 0x5BB

(1467) CGA28 (#0x5BB, RWN, init.

- int REG_MATES_DAC5_MK1_CGA29 = 0x5BC

(1468) CGA29 (#0x5BC, RWN, init.

- int REG_MATES_DAC5_MK1_CGA30 = 0x5BD

(1469) CGA30 (#0x5BD, RWN, init.

- int REG_MATES_DAC5_MK1_CGA31 = 0x5BE

(1470) CGA31 (#0x5BE, RWN, init.

- int REG_MATES_DAC5_MK1_CGA32 = 0x5BF

(1471) CGA32 (#0x5BF, RWN, init.

- int REG_MATES_DAC5_MK1_CGA33 = 0x5C0

(1472) CGA33 (#0x5C0, RWN, init.

- int REG_MATES_DAC5_MK1_CGA34 = 0x5C1

(1473) CGA34 (#0x5C1, RWN, init.

- int REG_MATES_DAC5_MK1_CGA35 = 0x5C2

(1474) CGA35 (#0x5C2, RWN, init.

- int REG_MATES_DAC5_MK1_CGA36 = 0x5C3

(1475) CGA36 (#0x5C3, RWN, init.

- int REG_MATES_DAC5_MK1_CGA37 = 0x5C4

(1476) CGA37 (#0x5C4, RWN, init.

- int REG_MATES_DAC5_MK1_CGA38 = 0x5C5

(1477) CGA38 (#0x5C5, RWN, init.

- int REG_MATES_DAC5_MK1_CGA39 = 0x5C6

(1478) CGA39 (#0x5C6, RWN, init.

- int REG_MATES_DAC5_MK1_CGA40 = 0x5C7

(1479) CGA40 (#0x5C7, RWN, init.

- int REG_MATES_DAC5_MK1_BMIN = 0x5C8

(1480) BMIN (#0x5C8, RWN, init.

- int REG_MATES_DAC5_MK1_BMAX = 0x5C9

(1481) BMAX (#0x5C9, RWN, init.

- int REG_MATES_DAC5_MK1_LCT = 0x5CA

(1482) LCT (#0x5CA, RWN, init.

- int REG_MATES_DAC5_MK1_CANID = 0x5CB

(1483) CANID (#0x5CB, RWN, init.

- int REG_MATES_DIOX_MK1_ODEF = 0x400

(1024) ODEF (#0x400, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL01 = 0x401

(1025) OCTRL01 (#0x401, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL02 = 0x402

(1026) OCTRL02 (#0x402, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL03 = 0x403

(1027) OCTRL03 (#0x403, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL04 = 0x404

(1028) OCTRL04 (#0x404, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL05 = 0x405

(1029) OCTRL05 (#0x405, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL06 = 0x406

(1030) OCTRL06 (#0x406, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL07 = 0x407

(1031) OCTRL07 (#0x407, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL08 = 0x408

(1032) OCTRL08 (#0x408, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL09 = 0x409

(1033) OCTRL09 (#0x409, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL10 = 0x40A

(1034) OCTRL10 (#0x40A, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL11 = 0x40B

(1035) OCTRL11 (#0x40B, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL12 = 0x40C

(1036) OCTRL12 (#0x40C, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL13 = 0x40D

(1037) OCTRL13 (#0x40D, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL14 = 0x40E

(1038) OCTRL14 (#0x40E, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL15 = 0x40F

(1039) OCTRL15 (#0x40F, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL16 = 0x410

(1040) OCTRL16 (#0x410, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL17 = 0x411

(1041) OCTRL17 (#0x411, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL18 = 0x412

(1042) OCTRL18 (#0x412, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL19 = 0x413

(1043) OCTRL19 (#0x413, RWN, init.

- int REG_MATES_DIOX_MK1_OCTRL20 = 0x414

(1044) OCTRL20 (#0x414, RWN, init.

- int REG_MATES_DIOX_MK1_ORP = 0x415

(1045) ORP (#0x415, RWN, init.

- int REG_MATES_DIOX_MK1_ORRT = 0x416

(1046) ORRT (#0x416, RO, init.

- int REG_MATES_DIOX_MK1_CANID = 0x417

(1047) CANID (#0x417, RWN, init.

- int REG_MATES_ADC5_MK1_COF01 = 0x600

(1536) COF01 (#0x600, RWN, init.

- int REG_MATES_ADC5_MK1_COF02 = 0x601

(1537) COF02 (#0x601, RWN, init.

- int REG_MATES_ADC5_MK1_COF03 = 0x602

(1538) COF03 (#0x602, RWN, init.

- int REG_MATES_ADC5_MK1_COF04 = 0x603

(1539) COF04 (#0x603, RWN, init.

- int REG_MATES_ADC5_MK1_COF05 = 0x604

(1540) COF05 (#0x604, RWN, init.

- int REG_MATES_ADC5_MK1_COF06 = 0x605

(1541) COF06 (#0x605, RWN, init.

- int REG_MATES_ADC5_MK1_COF07 = 0x606

(1542) COF07 (#0x606, RWN, init.

- int REG_MATES_ADC5_MK1_COF08 = 0x607

(1543) COF08 (#0x607, RWN, init.

- int REG_MATES_ADC5_MK1_COF09 = 0x608

(1544) COF09 (#0x608, RWN, init.

- int REG_MATES_ADC5_MK1_COF10 = 0x609

(1545) COF10 (#0x609, RWN, init.

- int REG_MATES_ADC5_MK1_COF11 = 0x60A

(1546) COF11 (#0x60A, RWN, init.

- int REG_MATES_ADC5_MK1_COF12 = 0x60B

(1547) COF12 (#0x60B, RWN, init.

- int REG_MATES_ADC5_MK1_COF13 = 0x60C

(1548) COF13 (#0x60C, RWN, init.

- int REG_MATES_ADC5_MK1_COF14 = 0x60D

(1549) COF14 (#0x60D, RWN, init.

- int REG_MATES_ADC5_MK1_COF15 = 0x60E

(1550) COF15 (#0x60E, RWN, init.

- int REG_MATES_ADC5_MK1_COF16 = 0x60F

(1551) COF16 (#0x60F, RWN, init.

- int REG_MATES_ADC5_MK1_COF17 = 0x610

(1552) COF17 (#0x610, RWN, init.

- int REG_MATES_ADC5_MK1_COF18 = 0x611

(1553) COF18 (#0x611, RWN, init.

- int REG_MATES_ADC5_MK1_COF19 = 0x612

(1554) COF19 (#0x612, RWN, init.

- int REG_MATES_ADC5_MK1_COF20 = 0x613

(1555) COF20 (#0x613, RWN, init.

- int REG_MATES_ADC5_MK1_COF21 = 0x614

(1556) COF21 (#0x614, RWN, init.

- int REG_MATES_ADC5_MK1_COF22 = 0x615

(1557) COF22 (#0x615, RWN, init.

- int REG_MATES_ADC5_MK1_COF23 = 0x616

(1558) COF23 (#0x616, RWN, init.

- int REG_MATES_ADC5_MK1_COF24 = 0x617

(1559) COF24 (#0x617, RWN, init.

- int REG_MATES_ADC5_MK1_COF25 = 0x618

(1560) COF25 (#0x618, RWN, init.

- int REG_MATES_ADC5_MK1_COF26 = 0x619

(1561) COF26 (#0x619, RWN, init.

- int REG_MATES_ADC5_MK1_COF27 = 0x61A

(1562) COF27 (#0x61A, RWN, init.

- int REG_MATES_ADC5_MK1_COF28 = 0x61B

(1563) COF28 (#0x61B, RWN, init.

- int REG_MATES_ADC5_MK1_COF29 = 0x61C

(1564) COF29 (#0x61C, RWN, init.

- int REG_MATES_ADC5_MK1_COF30 = 0x61D

(1565) COF30 (#0x61D, RWN, init.

- int REG_MATES_ADC5_MK1_COF31 = 0x61E

(1566) COF31 (#0x61E, RWN, init.

- int REG_MATES_ADC5_MK1_COF32 = 0x61F

(1567) COF32 (#0x61F, RWN, init.

- int REG_MATES_ADC5_MK1_COF33 = 0x620

(1568) COF33 (#0x620, RWN, init.

- int REG_MATES_ADC5_MK1_COF34 = 0x621

(1569) COF34 (#0x621, RWN, init.

- int REG_MATES_ADC5_MK1_COF35 = 0x622

(1570) COF35 (#0x622, RWN, init.

- int REG_MATES_ADC5_MK1_COF36 = 0x623

(1571) COF36 (#0x623, RWN, init.

- int REG_MATES_ADC5_MK1_COF37 = 0x624

(1572) COF37 (#0x624, RWN, init.

- int REG_MATES_ADC5_MK1_COF38 = 0x625

(1573) COF38 (#0x625, RWN, init.

- int REG_MATES_ADC5_MK1_COF39 = 0x626

(1574) COF39 (#0x626, RWN, init.

- int REG_MATES_ADC5_MK1_COF40 = 0x627

(1575) COF40 (#0x627, RWN, init.

- int REG_MATES_ADC5_MK1_CGA01 = 0x628

(1576) CGA01 (#0x628, RWN, init.

- int REG_MATES_ADC5_MK1_CGA02 = 0x629

(1577) CGA02 (#0x629, RWN, init.

- int REG_MATES_ADC5_MK1_CGA03 = 0x62A

(1578) CGA03 (#0x62A, RWN, init.

- int REG_MATES_ADC5_MK1_CGA04 = 0x62B

(1579) CGA04 (#0x62B, RWN, init.

- int REG_MATES_ADC5_MK1_CGA05 = 0x62C

(1580) CGA05 (#0x62C, RWN, init.

- int REG_MATES_ADC5_MK1_CGA06 = 0x62D

(1581) CGA06 (#0x62D, RWN, init.

- int REG_MATES_ADC5_MK1_CGA07 = 0x62E

(1582) CGA07 (#0x62E, RWN, init.

- int REG_MATES_ADC5_MK1_CGA08 = 0x62F

(1583) CGA08 (#0x62F, RWN, init.

- int REG_MATES_ADC5_MK1_CGA09 = 0x630

(1584) CGA09 (#0x630, RWN, init.

- int REG_MATES_ADC5_MK1_CGA10 = 0x631

(1585) CGA10 (#0x631, RWN, init.

- int REG_MATES_ADC5_MK1_CGA11 = 0x632

(1586) CGA11 (#0x632, RWN, init.

- int REG_MATES_ADC5_MK1_CGA12 = 0x633

(1587) CGA12 (#0x633, RWN, init.

- int REG_MATES_ADC5_MK1_CGA13 = 0x634

(1588) CGA13 (#0x634, RWN, init.

- int REG_MATES_ADC5_MK1_CGA14 = 0x635

(1589) CGA14 (#0x635, RWN, init.

- int REG_MATES_ADC5_MK1_CGA15 = 0x636

(1590) CGA15 (#0x636, RWN, init.

- int REG_MATES_ADC5_MK1_CGA16 = 0x637

(1591) CGA16 (#0x637, RWN, init.

- int REG_MATES_ADC5_MK1_CGA17 = 0x638

(1592) CGA17 (#0x638, RWN, init.

- int REG_MATES_ADC5_MK1_CGA18 = 0x639

(1593) CGA18 (#0x639, RWN, init.

- int REG_MATES_ADC5_MK1_CGA19 = 0x63A

(1594) CGA19 (#0x63A, RWN, init.

- int REG_MATES_ADC5_MK1_CGA20 = 0x63B

(1595) CGA20 (#0x63B, RWN, init.

- int REG_MATES_ADC5_MK1_CGA21 = 0x63C

(1596) CGA21 (#0x63C, RWN, init.

- int REG_MATES_ADC5_MK1_CGA22 = 0x63D

(1597) CGA22 (#0x63D, RWN, init.

- int REG_MATES_ADC5_MK1_CGA23 = 0x63E

(1598) CGA23 (#0x63E, RWN, init.

- int REG_MATES_ADC5_MK1_CGA24 = 0x63F

(1599) CGA24 (#0x63F, RWN, init.

- int REG_MATES_ADC5_MK1_CGA25 = 0x640

(1600) CGA25 (#0x640, RWN, init.

- int REG_MATES_ADC5_MK1_CGA26 = 0x641

(1601) CGA26 (#0x641, RWN, init.

- int REG_MATES_ADC5_MK1_CGA27 = 0x642

(1602) CGA27 (#0x642, RWN, init.

- int REG_MATES_ADC5_MK1_CGA28 = 0x643

(1603) CGA28 (#0x643, RWN, init.

- int REG_MATES_ADC5_MK1_CGA29 = 0x644

(1604) CGA29 (#0x644, RWN, init.

- int REG_MATES_ADC5_MK1_CGA30 = 0x645

(1605) CGA30 (#0x645, RWN, init.

- int REG_MATES_ADC5_MK1_CGA31 = 0x646

(1606) CGA31 (#0x646, RWN, init.

- int REG_MATES_ADC5_MK1_CGA32 = 0x647

(1607) CGA32 (#0x647, RWN, init.

- int REG_MATES_ADC5_MK1_CGA33 = 0x648

(1608) CGA33 (#0x648, RWN, init.

- int REG_MATES_ADC5_MK1_CGA34 = 0x649

(1609) CGA34 (#0x649, RWN, init.

- int REG_MATES_ADC5_MK1_CGA35 = 0x64A

(1610) CGA35 (#0x64A, RWN, init.

- int REG_MATES_ADC5_MK1_CGA36 = 0x64B

(1611) CGA36 (#0x64B, RWN, init.

- int REG_MATES_ADC5_MK1_CGA37 = 0x64C

(1612) CGA37 (#0x64C, RWN, init.

- int REG_MATES_ADC5_MK1_CGA38 = 0x64D

(1613) CGA38 (#0x64D, RWN, init.

- int REG_MATES_ADC5_MK1_CGA39 = 0x64E

(1614) CGA39 (#0x64E, RWN, init.

- int REG_MATES_ADC5_MK1_CGA40 = 0x64F

(1615) CGA40 (#0x64F, RWN, init.

- int REG_MATES_ADC5_MK1_LCT = 0x650

(1616) LCT (#0x650, RWN, init.

- int REG_MATES_ADC5_MK1_CANID = 0x651

(1617) CANID (#0x651, RWN, init.

- int REG_MATES_UCC_MK1_ODEF = 0x300

  (768) ODEF (#0x300, RWN, init.
- int REG_MATES_UCC_MK1_ODIS = 0x301

  (769) ODIS (#0x301, RW, init.
- int REG_MATES_UCC_MK1_PRD01 = 0x302

  (770) PRD01 (#0x302, RW, init.
- int REG_MATES_UCC_MK1_PRD02 = 0x303

  (771) PRD02 (#0x303, RW, init.
- int REG_MATES_UCC_MK1_PRD03 = 0x304

  (772) PRD03 (#0x304, RW, init.
- int REG_MATES_UCC_MK1_PRD04 = 0x305

  (773) PRD04 (#0x305, RW, init.
- int REG_MATES_UCC_MK1_PRD05 = 0x306

  (774) PRD05 (#0x306, RW, init.
- int REG_MATES_UCC_MK1_PRD06 = 0x307

  (775) PRD06 (#0x307, RW, init.
- int REG_MATES_UCC_MK1_PRD07 = 0x308

  (776) PRD07 (#0x308, RW, init.
- int REG_MATES_UCC_MK1_PRD08 = 0x309

  (777) PRD08 (#0x309, RW, init.
- int REG_MATES_UCC_MK1_PRD09 = 0x30A

  (778) PRD09 (#0x30A, RW, init.
- int REG_MATES_UCC_MK1_PRD10 = 0x30B

  (779) PRD10 (#0x30B, RW, init.
- int REG_MATES_UCC_MK1_PRD11 = 0x30C

  (780) PRD11 (#0x30C, RW, init.
- int REG_MATES_UCC_MK1_PRD12 = 0x30D

  (781) PRD12 (#0x30D, RW, init.
- int REG_MATES_UCC_MK1_PRD13 = 0x30E

  (782) PRD13 (#0x30E, RW, init.
- int REG_MATES_UCC_MK1_PRD14 = 0x30F

  (783) PRD14 (#0x30F, RW, init.
- int REG_MATES_UCC_MK1_TRISE01 = 0x310

  (784) TRISE01 (#0x310, RW, init.
- int REG_MATES_UCC_MK1_TRISE02 = 0x311

  (785) TRISE02 (#0x311, RW, init.
- int REG_MATES_UCC_MK1_TRISE03 = 0x312

  (786) TRISE03 (#0x312, RW, init.
- int REG_MATES_UCC_MK1_TRISE04 = 0x313

  (787) TRISE04 (#0x313, RW, init.
- int REG_MATES_UCC_MK1_TRISE05 = 0x314

(788) TRISE05 (#0x314, RW, init.

- int REG_MATES_UCC_MK1_TRISE06 = 0x315

(789) TRISE06 (#0x315, RW, init.

- int REG_MATES_UCC_MK1_TRISE07 = 0x316

(790) TRISE07 (#0x316, RW, init.

- int REG_MATES_UCC_MK1_TRISE08 = 0x317

(791) TRISE08 (#0x317, RW, init.

- int REG_MATES_UCC_MK1_TRISE09 = 0x318

(792) TRISE09 (#0x318, RW, init.

- int REG_MATES_UCC_MK1_TRISE10 = 0x319

(793) TRISE10 (#0x319, RW, init.

- int REG_MATES_UCC_MK1_TRISE11 = 0x31A

(794) TRISE11 (#0x31A, RW, init.

- int REG_MATES_UCC_MK1_TRISE12 = 0x31B

(795) TRISE12 (#0x31B, RW, init.

- int REG_MATES_UCC_MK1_TRISE13 = 0x31C

(796) TRISE13 (#0x31C, RW, init.

- int REG_MATES_UCC_MK1_TRISE14 = 0x31D

(797) TRISE14 (#0x31D, RW, init.

- int REG_MATES_UCC_MK1_TFALL01 = 0x31E

(798) TFALL01 (#0x31E, RW, init.

- int REG_MATES_UCC_MK1_TFALL02 = 0x31F

(799) TFALL02 (#0x31F, RW, init.

- int REG_MATES_UCC_MK1_TFALL03 = 0x320

(800) TFALL03 (#0x320, RW, init.

- int REG_MATES_UCC_MK1_TFALL04 = 0x321

(801) TFALL04 (#0x321, RW, init.

- int REG_MATES_UCC_MK1_TFALL05 = 0x322

(802) TFALL05 (#0x322, RW, init.

- int REG_MATES_UCC_MK1_TFALL06 = 0x323

(803) TFALL06 (#0x323, RW, init.

- int REG_MATES_UCC_MK1_TFALL07 = 0x324

(804) TFALL07 (#0x324, RW, init.

- int REG_MATES_UCC_MK1_TFALL08 = 0x325

(805) TFALL08 (#0x325, RW, init.

- int REG_MATES_UCC_MK1_TFALL09 = 0x326

(806) TFALL09 (#0x326, RW, init.

- int REG_MATES_UCC_MK1_TFALL10 = 0x327

(807) TFALL10 (#0x327, RW, init.

- int REG_MATES_UCC_MK1_TFALL11 = 0x328

(808) TFALL11 (#0x328, RW, init.

- int REG_MATES_UCC_MK1_TFALL12 = 0x329

(809) TFALL12 (#0x329, RW, init.

- int REG_MATES_UCC_MK1_TFALL13 = 0x32A

(810) TFALL13 (#0x32A, RW, init.

- int REG_MATES_UCC_MK1_TFALL14 = 0x32B

(811) TFALL14 (#0x32B, RW, init.

- int REG_MATES_UCC_MK1_ORP = 0x32C

(812) ORP (#0x32C, RWN, init.

- int REG_MATES_UCC_MK1_ORRT = 0x32D

(813) ORRT (#0x32D, RO, init.

- int REG_MATES_UCC_MK1_CANID = 0x32E

(814) CANID (#0x32E, RWN, init.

- int REG_MATES_UCC_MK1_PWMD = 0x32F

(815) PWMD (#0x32F, RW, init.

- int REG_MATES_UCC_MK1_ALCDB01 = 0x330

(816) ALCDB01 (#0x330, RW, init.

- int REG_MATES_UCC_MK1_ALCDB02 = 0x331

(817) ALCDB02 (#0x331, RW, init.

- int REG_MATES_UCC_MK1_ALCDB03 = 0x332

(818) ALCDB03 (#0x332, RW, init.

- int REG_MATES_UCC_MK1_ALCDB04 = 0x333

(819) ALCDB04 (#0x333, RW, init.

- int REG_MATES_UCC_MK1_ALCDB05 = 0x334

(820) ALCDB05 (#0x334, RW, init.

- int REG_MATES_UCC_MK1_ALCDB06 = 0x335

(821) ALCDB06 (#0x335, RW, init.

- int REG_MATES_UCC_MK1_ALCDB07 = 0x336

(822) ALCDB07 (#0x336, RW, init.

- int REG_MATES_UCC_MK1_ALCDB08 = 0x337

(823) ALCDB08 (#0x337, RW, init.

A.15.1 Detailed Description

MATES registers numbers.

See `MATES User's manual` for registers documentation.

A.15.2 Field Documentation

int REG_COMMON_TEST = 0x0 `[static]`

(0) TEST (#0x0, RWN, init.

: 0xFF, min.: 0x0, max.: 0xFFFFFFFF)

int REG_COMMON_DIR = 0x1 [static]

(1) DIR (#0x1, RO, init.
: 0x0)

int REG_COMMON_BIR = 0x2 [static]

(2) BIR (#0x2, RO, init.
: 0x0)

int REG_COMMON_FIR = 0x3 [static]

(3) FIR (#0x3, RO, init.
: 0x0)

int REG_COMMON_SR = 0x4 [static]

(4) SR (#0x4, RO, init.
: 0x0)

int REG_COMMON_CR = 0x5 [static]

(5) CR (#0x5, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_COMMON_TUT = 0x6 [static]

(6) TUT (#0x6, RO, init.
: 0x0)

int REG_COMMON_CUT = 0x7 [static]

(7) CUT (#0x7, RO, init.
: 0x0)

int REG_COMMON_FUT = 0x8 [static]

(8) FUT (#0x8, RO, init.
: 0x0)

int REG_COMMON_DPD = 0x9 [static]

(9) DPD (#0x9, RO, init.
: 0x0)

int REG_COMMON_PUC = 0xA [static]

(10) PUC (#0xA, RO, init.
: 0x0)

int REG_COMMON_HIR = 0xB [static]

(11) HIR (#0xB, RO, init.
: 0x0)

int REG_COMMON_CAPS01 = 0xC [static]

(12) CAPS01 (#0xC, RO, init.
: 0x0)

int REG_COMMON_CAPS02 = 0xD [static]

(13) CAPS02 (#0xD, RO, init.
: 0x0)

int REG_MATES_DIO3_MK1_ODEF = 0x200 [static]

(512) ODEF (#0x200, RWN, init.
: 0x0, min.: 0x0, max.: 0x800FFFFF)

int REG_MATES_DIO3_MK1_ODIS = 0x201 [static]

(513) ODIS (#0x201, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFF)

int REG_MATES_DIO3_MK1_PRD01 = 0x202 [static]

(514) PRD01 (#0x202, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD02 = 0x203 [static]

(515) PRD02 (#0x203, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD03 = 0x204 [static]

(516) PRD03 (#0x204, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD04 = 0x205 [static]

(517) PRD04 (#0x205, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD05 = 0x206 [static]

(518) PRD05 (#0x206, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD06 = 0x207 [static]

(519) PRD06 (#0x207, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD07 = 0x208 [static]

(520) PRD07 (#0x208, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD08 = 0x209 [static]

(521) PRD08 (#0x209, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD09 = 0x20A [static]

(522) PRD09 (#0x20A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD10 = 0x20B [static]

(523) PRD10 (#0x20B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD11 = 0x20C [static]

(524) PRD11 (#0x20C, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD12 = 0x20D [static]

(525) PRD12 (#0x20D, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD13 = 0x20E [static]

(526) PRD13 (#0x20E, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD14 = 0x20F [static]

(527) PRD14 (#0x20F, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD15 = 0x210 [static]

(528) PRD15 (#0x210, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD16 = 0x211 [static]

(529) PRD16 (#0x211, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD17 = 0x212 [static]

(530) PRD17 (#0x212, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD18 = 0x213 [static]

(531) PRD18 (#0x213, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD19 = 0x214 [static]

(532) PRD19 (#0x214, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_PRD20 = 0x215 [static]

(533) PRD20 (#0x215, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE01 = 0x216 [static]

(534) TRISE01 (#0x216, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE02 = 0x217 [static]

(535) TRISE02 (#0x217, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE03 = 0x218 [static]

(536) TRISE03 (#0x218, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE04 = 0x219 [static]

(537) TRISE04 (#0x219, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE05 = 0x21A [static]

(538) TRISE05 (#0x21A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE06 = 0x21B [static]

(539) TRISE06 (#0x21B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE07 = 0x21C [static]

(540) TRISE07 (#0x21C, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE08 = 0x21D [static]

(541) TRISE08 (#0x21D, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE09 = 0x21E [static]

(542) TRISE09 (#0x21E, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE10 = 0x21F [static]

(543) TRISE10 (#0x21F, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE11 = 0x220 [static]

(544) TRISE11 (#0x220, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE12 = 0x221 [static]

(545) TRISE12 (#0x221, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE13 = 0x222 [static]

(546) TRISE13 (#0x222, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE14 = 0x223 [static]

(547) TRISE14 (#0x223, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE15 = 0x224 [static]

(548) TRISE15 (#0x224, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE16 = 0x225 [static]

(549) TRISE16 (#0x225, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE17 = 0x226 [static]

(550) TRISE17 (#0x226, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE18 = 0x227 [static]

(551) TRISE18 (#0x227, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE19 = 0x228 [static]

(552) TRISE19 (#0x228, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TRISE20 = 0x229 [static]

(553) TRISE20 (#0x229, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL01 = 0x22A [static]

(554) TFALL01 (#0x22A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL02 = 0x22B [static]

(555) TFALL02 (#0x22B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL03 = 0x22C [static]

(556) TFALL03 (#0x22C, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL04 = 0x22D [static]

(557) TFALL04 (#0x22D, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL05 = 0x22E [static]

(558) TFALL05 (#0x22E, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL06 = 0x22F [static]

(559) TFALL06 (#0x22F, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL07 = 0x230 [static]

(560) TFALL07 (#0x230, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL08 = 0x231 [static]

(561) TFALL08 (#0x231, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL09 = 0x232 [static]

(562) TFALL09 (#0x232, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL10 = 0x233 [static]

(563) TFALL10 (#0x233, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL11 = 0x234 [static]

(564) TFALL11 (#0x234, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL12 = 0x235 [static]

(565) TFALL12 (#0x235, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL13 = 0x236 [static]

(566) TFALL13 (#0x236, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL14 = 0x237 [static]

(567) TFALL14 (#0x237, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL15 = 0x238 [static]

(568) TFALL15 (#0x238, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL16 = 0x239 [static]

(569) TFALL16 (#0x239, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL17 = 0x23A [static]

(570) TFALL17 (#0x23A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL18 = 0x23B [static]

(571) TFALL18 (#0x23B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL19 = 0x23C [static]

(572) TFALL19 (#0x23C, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_TFALL20 = 0x23D [static]

(573) TFALL20 (#0x23D, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_ORP = 0x23E [static]

(574) ORP (#0x23E, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIO3_MK1_ORRT = 0x23F [static]

(575) ORRT (#0x23F, RO, init.
: 0x0)

int REG_MATES_DIO3_MK1_PSEL = 0x240 [static]

(576) PSEL (#0x240, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFF)

int REG_MATES_DIO3_MK1_CANID = 0x241 [static]

(577) CANID (#0x241, RWN, init.
: 20, min.: 20, max.: 29)

int REG_MATES_DAC5_MK1_DEF01 = 0x500 [static]

(1280) DEF01 (#0x500, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF02 = 0x501 [static]

(1281) DEF02 (#0x501, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF03 = 0x502 [static]

(1282) DEF03 (#0x502, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF04 = 0x503 [static]

(1283) DEF04 (#0x503, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF05 = 0x504 [static]

(1284) DEF05 (#0x504, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF06 = 0x505 [static]

(1285) DEF06 (#0x505, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF07 = 0x506 [static]

(1286) DEF07 (#0x506, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF08 = 0x507 [static]

(1287) DEF08 (#0x507, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF09 = 0x508 [static]

(1288) DEF09 (#0x508, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF10 = 0x509 [static]

(1289) DEF10 (#0x509, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF11 = 0x50A [static]

(1290) DEF11 (#0x50A, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF12 = 0x50B [static]

(1291) DEF12 (#0x50B, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF13 = 0x50C [static]

(1292) DEF13 (#0x50C, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF14 = 0x50D [static]

(1293) DEF14 (#0x50D, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF15 = 0x50E [static]

(1294) DEF15 (#0x50E, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF16 = 0x50F [static]

(1295) DEF16 (#0x50F, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF17 = 0x510 [static]

(1296) DEF17 (#0x510, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF18 = 0x511 [static]

(1297) DEF18 (#0x511, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF19 = 0x512 [static]

(1298) DEF19 (#0x512, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF20 = 0x513 [static]

(1299) DEF20 (#0x513, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF21 = 0x514 [static]

(1300) DEF21 (#0x514, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF22 = 0x515 [static]

(1301) DEF22 (#0x515, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF23 = 0x516 [static]

(1302) DEF23 (#0x516, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF24 = 0x517 [static]

(1303) DEF24 (#0x517, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF25 = 0x518 [static]

(1304) DEF25 (#0x518, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF26 = 0x519 [static]

(1305) DEF26 (#0x519, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF27 = 0x51A [static]

(1306) DEF27 (#0x51A, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF28 = 0x51B [static]

(1307) DEF28 (#0x51B, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF29 = 0x51C [static]

(1308) DEF29 (#0x51C, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF30 = 0x51D [static]

(1309) DEF30 (#0x51D, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF31 = 0x51E [static]

(1310) DEF31 (#0x51E, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF32 = 0x51F [static]

(1311) DEF32 (#0x51F, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF33 = 0x520 [static]

(1312) DEF33 (#0x520, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF34 = 0x521 [static]

(1313) DEF34 (#0x521, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF35 = 0x522 [static]

(1314) DEF35 (#0x522, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF36 = 0x523 [static]

(1315) DEF36 (#0x523, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF37 = 0x524 [static]

(1316) DEF37 (#0x524, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF38 = 0x525 [static]

(1317) DEF38 (#0x525, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF39 = 0x526 [static]

(1318) DEF39 (#0x526, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_DEF40 = 0x527 [static]

(1319) DEF40 (#0x527, RWN, init.
: 0.1, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN01 = 0x528 [static]

(1320) MIN01 (#0x528, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN02 = 0x529 [static]

(1321) MIN02 (#0x529, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN03 = 0x52A [static]

(1322) MIN03 (#0x52A, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN04 = 0x52B [static]

(1323) MIN04 (#0x52B, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN05 = 0x52C [static]

(1324) MIN05 (#0x52C, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN06 = 0x52D [static]

(1325) MIN06 (#0x52D, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN07 = 0x52E [static]

(1326) MIN07 (#0x52E, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN08 = 0x52F [static]

(1327) MIN08 (#0x52F, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN09 = 0x530 [static]

(1328) MIN09 (#0x530, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN10 = 0x531 [static]

(1329) MIN10 (#0x531, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN11 = 0x532 [static]

(1330) MIN11 (#0x532, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN12 = 0x533 [static]

(1331) MIN12 (#0x533, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN13 = 0x534 [static]

(1332) MIN13 (#0x534, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN14 = 0x535 [static]

(1333) MIN14 (#0x535, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN15 = 0x536 [static]

(1334) MIN15 (#0x536, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN16 = 0x537 [static]

(1335) MIN16 (#0x537, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN17 = 0x538 [static]

(1336) MIN17 (#0x538, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN18 = 0x539 [static]

(1337) MIN18 (#0x539, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN19 = 0x53A [static]

(1338) MIN19 (#0x53A, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN20 = 0x53B [static]

(1339) MIN20 (#0x53B, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN21 = 0x53C [static]

(1340) MIN21 (#0x53C, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN22 = 0x53D [static]

(1341) MIN22 (#0x53D, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN23 = 0x53E [static]

(1342) MIN23 (#0x53E, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN24 = 0x53F [static]

(1343) MIN24 (#0x53F, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN25 = 0x540 [static]

(1344) MIN25 (#0x540, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN26 = 0x541 [static]

(1345) MIN26 (#0x541, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN27 = 0x542 [static]

(1346) MIN27 (#0x542, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN28 = 0x543 [static]

(1347) MIN28 (#0x543, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN29 = 0x544 [static]

(1348) MIN29 (#0x544, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN30 = 0x545 [static]

(1349) MIN30 (#0x545, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN31 = 0x546 [static]

(1350) MIN31 (#0x546, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN32 = 0x547 [static]

(1351) MIN32 (#0x547, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN33 = 0x548 [static]

(1352) MIN33 (#0x548, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN34 = 0x549 [static]

(1353) MIN34 (#0x549, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN35 = 0x54A [static]

(1354) MIN35 (#0x54A, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN36 = 0x54B [static]

(1355) MIN36 (#0x54B, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN37 = 0x54C [static]

(1356) MIN37 (#0x54C, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN38 = 0x54D [static]

(1357) MIN38 (#0x54D, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN39 = 0x54E [static]

(1358) MIN39 (#0x54E, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MIN40 = 0x54F [static]

(1359) MIN40 (#0x54F, RWN, init.
: 0.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX01 = 0x550 [static]

(1360) MAX01 (#0x550, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX02 = 0x551 [static]

(1361) MAX02 (#0x551, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX03 = 0x552 [static]

(1362) MAX03 (#0x552, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX04 = 0x553 [static]

(1363) MAX04 (#0x553, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX05 = 0x554 [static]

(1364) MAX05 (#0x554, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX06 = 0x555 [static]

(1365) MAX06 (#0x555, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX07 = 0x556 [static]

(1366) MAX07 (#0x556, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX08 = 0x557 [static]

(1367) MAX08 (#0x557, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX09 = 0x558 [static]

(1368) MAX09 (#0x558, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX10 = 0x559 [static]

(1369) MAX10 (#0x559, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX11 = 0x55A [static]

(1370) MAX11 (#0x55A, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX12 = 0x55B [static]

(1371) MAX12 (#0x55B, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX13 = 0x55C [static]

(1372) MAX13 (#0x55C, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX14 = 0x55D [static]

(1373) MAX14 (#0x55D, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX15 = 0x55E [static]

(1374) MAX15 (#0x55E, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX16 = 0x55F [static]

(1375) MAX16 (#0x55F, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX17 = 0x560 [static]

(1376) MAX17 (#0x560, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX18 = 0x561 [static]

(1377) MAX18 (#0x561, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX19 = 0x562 [static]

(1378) MAX19 (#0x562, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX20 = 0x563 [static]

(1379) MAX20 (#0x563, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX21 = 0x564 [static]

(1380) MAX21 (#0x564, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX22 = 0x565 [static]

(1381) MAX22 (#0x565, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX23 = 0x566 [static]

(1382) MAX23 (#0x566, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX24 = 0x567 [static]

(1383) MAX24 (#0x567, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX25 = 0x568 [static]

(1384) MAX25 (#0x568, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX26 = 0x569 [static]

(1385) MAX26 (#0x569, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX27 = 0x56A [static]

(1386) MAX27 (#0x56A, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX28 = 0x56B [static]

(1387) MAX28 (#0x56B, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX29 = 0x56C [static]

(1388) MAX29 (#0x56C, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX30 = 0x56D [static]

(1389) MAX30 (#0x56D, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX31 = 0x56E [static]

(1390) MAX31 (#0x56E, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX32 = 0x56F [static]

(1391) MAX32 (#0x56F, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX33 = 0x570 [static]

(1392) MAX33 (#0x570, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX34 = 0x571 [static]

(1393) MAX34 (#0x571, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX35 = 0x572 [static]

(1394) MAX35 (#0x572, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX36 = 0x573 [static]

(1395) MAX36 (#0x573, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX37 = 0x574 [static]

(1396) MAX37 (#0x574, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX38 = 0x575 [static]

(1397) MAX38 (#0x575, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX39 = 0x576 [static]

(1398) MAX39 (#0x576, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_MAX40 = 0x577 [static]

(1399) MAX40 (#0x577, RWN, init.
: 3.0, min.: 0.0, max.: 5.0)

int REG_MATES_DAC5_MK1_COF01 = 0x578 [static]

(1400) COF01 (#0x578, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF02 = 0x579 [static]

(1401) COF02 (#0x579, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF03 = 0x57A [static]

(1402) COF03 (#0x57A, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF04 = 0x57B [static]

(1403) COF04 (#0x57B, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF05 = 0x57C [static]

(1404) COF05 (#0x57C, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF06 = 0x57D [static]

(1405) COF06 (#0x57D, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF07 = 0x57E [static]

(1406) COF07 (#0x57E, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF08 = 0x57F [static]

(1407) COF08 (#0x57F, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF09 = 0x580 [static]

(1408) COF09 (#0x580, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF10 = 0x581 [static]

(1409) COF10 (#0x581, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF11 = 0x582 [static]

(1410) COF11 (#0x582, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF12 = 0x583 [static]

(1411) COF12 (#0x583, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF13 = 0x584 [static]

(1412) COF13 (#0x584, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF14 = 0x585 [static]

(1413) COF14 (#0x585, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF15 = 0x586 [static]

(1414) COF15 (#0x586, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF16 = 0x587 [static]

(1415) COF16 (#0x587, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF17 = 0x588 [static]

(1416) COF17 (#0x588, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF18 = 0x589 [static]

(1417) COF18 (#0x589, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF19 = 0x58A [static]

(1418) COF19 (#0x58A, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF20 = 0x58B [static]

(1419) COF20 (#0x58B, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF21 = 0x58C [static]

(1420) COF21 (#0x58C, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF22 = 0x58D [static]

(1421) COF22 (#0x58D, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF23 = 0x58E [static]

(1422) COF23 (#0x58E, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF24 = 0x58F [static]

(1423) COF24 (#0x58F, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF25 = 0x590 [static]

(1424) COF25 (#0x590, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF26 = 0x591 [static]

(1425) COF26 (#0x591, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF27 = 0x592 [static]

(1426) COF27 (#0x592, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF28 = 0x593 [static]

(1427) COF28 (#0x593, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF29 = 0x594 [static]

(1428) COF29 (#0x594, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF30 = 0x595 [static]

(1429) COF30 (#0x595, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF31 = 0x596 [static]

(1430) COF31 (#0x596, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF32 = 0x597 [static]

(1431) COF32 (#0x597, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF33 = 0x598 [static]

(1432) COF33 (#0x598, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF34 = 0x599 [static]

(1433) COF34 (#0x599, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF35 = 0x59A [static]

(1434) COF35 (#0x59A, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF36 = 0x59B [static]

(1435) COF36 (#0x59B, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF37 = 0x59C [static]

(1436) COF37 (#0x59C, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF38 = 0x59D [static]

(1437) COF38 (#0x59D, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF39 = 0x59E [static]

(1438) COF39 (#0x59E, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_COF40 = 0x59F [static]

(1439) COF40 (#0x59F, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_DAC5_MK1_CGA01 = 0x5A0 [static]

(1440) CGA01 (#0x5A0, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA02 = 0x5A1 [static]

(1441) CGA02 (#0x5A1, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA03 = 0x5A2 [static]

(1442) CGA03 (#0x5A2, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA04 = 0x5A3 [static]

(1443) CGA04 (#0x5A3, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA05 = 0x5A4 [static]

(1444) CGA05 (#0x5A4, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA06 = 0x5A5 [static]

(1445) CGA06 (#0x5A5, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA07 = 0x5A6 [static]

(1446) CGA07 (#0x5A6, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA08 = 0x5A7 [static]

(1447) CGA08 (#0x5A7, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA09 = 0x5A8 [static]

(1448) CGA09 (#0x5A8, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA10 = 0x5A9 [static]

(1449) CGA10 (#0x5A9, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA11 = 0x5AA [static]

(1450) CGA11 (#0x5AA, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA12 = 0x5AB [static]

(1451) CGA12 (#0x5AB, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA13 = 0x5AC [static]

(1452) CGA13 (#0x5AC, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA14 = 0x5AD [static]

(1453) CGA14 (#0x5AD, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA15 = 0x5AE [static]

(1454) CGA15 (#0x5AE, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA16 = 0x5AF [static]

(1455) CGA16 (#0x5AF, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA17 = 0x5B0 [static]

(1456) CGA17 (#0x5B0, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA18 = 0x5B1 [static]

(1457) CGA18 (#0x5B1, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA19 = 0x5B2 [static]

(1458) CGA19 (#0x5B2, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA20 = 0x5B3 [static]

(1459) CGA20 (#0x5B3, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA21 = 0x5B4 [static]

(1460) CGA21 (#0x5B4, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA22 = 0x5B5 [static]

(1461) CGA22 (#0x5B5, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA23 = 0x5B6 [static]

(1462) CGA23 (#0x5B6, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA24 = 0x5B7 [static]

(1463) CGA24 (#0x5B7, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA25 = 0x5B8 [static]

(1464) CGA25 (#0x5B8, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA26 = 0x5B9 [static]

(1465) CGA26 (#0x5B9, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA27 = 0x5BA [static]

(1466) CGA27 (#0x5BA, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA28 = 0x5BB [static]

(1467) CGA28 (#0x5BB, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA29 = 0x5BC [static]

(1468) CGA29 (#0x5BC, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA30 = 0x5BD [static]

(1469) CGA30 (#0x5BD, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA31 = 0x5BE [static]

(1470) CGA31 (#0x5BE, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA32 = 0x5BF [static]

(1471) CGA32 (#0x5BF, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA33 = 0x5C0 [static]

(1472) CGA33 (#0x5C0, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA34 = 0x5C1 [static]

(1473) CGA34 (#0x5C1, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA35 = 0x5C2 [static]

(1474) CGA35 (#0x5C2, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA36 = 0x5C3 [static]

(1475) CGA36 (#0x5C3, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA37 = 0x5C4 [static]

(1476) CGA37 (#0x5C4, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA38 = 0x5C5 [static]

(1477) CGA38 (#0x5C5, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA39 = 0x5C6 [static]

(1478) CGA39 (#0x5C6, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_CGA40 = 0x5C7 [static]

(1479) CGA40 (#0x5C7, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_DAC5_MK1_BMIN = 0x5C8 [static]

(1480) BMIN (#0x5C8, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFF)

int REG_MATES_DAC5_MK1_BMAX = 0x5C9 [static]

(1481) BMAX (#0x5C9, RWN, init.
: 0x9A00, min.: 0x0, max.: 0xFFFF)

int REG_MATES_DAC5_MK1_LCT = 0x5CA [static]

(1482) LCT (#0x5CA, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DAC5_MK1_CANID = 0x5CB [static]

(1483) CANID (#0x5CB, RWN, init.
: 50, min.: 50, max.: 59)

int REG_MATES_DIOX_MK1_ODEF = 0x400 [static]

(1024) ODEF (#0x400, RWN, init.
: 0xC0000000, min.: 0x0, max.: 0xC00FFFFF)

int REG_MATES_DIOX_MK1_OCTRL01 = 0x401 [static]

(1025) OCTRL01 (#0x401, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL02 = 0x402 [static]

(1026) OCTRL02 (#0x402, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL03 = 0x403 [static]

(1027) OCTRL03 (#0x403, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL04 = 0x404 [static]

(1028) OCTRL04 (#0x404, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL05 = 0x405 [static]

(1029) OCTRL05 (#0x405, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL06 = 0x406 [static]

(1030) OCTRL06 (#0x406, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL07 = 0x407 [static]

(1031) OCTRL07 (#0x407, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL08 = 0x408 [static]

(1032) OCTRL08 (#0x408, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL09 = 0x409 [static]

(1033) OCTRL09 (#0x409, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL10 = 0x40A [static]

(1034) OCTRL10 (#0x40A, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL11 = 0x40B [static]

(1035) OCTRL11 (#0x40B, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL12 = 0x40C [static]

(1036) OCTRL12 (#0x40C, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL13 = 0x40D [static]

(1037) OCTRL13 (#0x40D, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL14 = 0x40E [static]

(1038) OCTRL14 (#0x40E, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL15 = 0x40F [static]

(1039) OCTRL15 (#0x40F, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL16 = 0x410 [static]

(1040) OCTRL16 (#0x410, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL17 = 0x411 [static]

(1041) OCTRL17 (#0x411, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL18 = 0x412 [static]

(1042) OCTRL18 (#0x412, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL19 = 0x413 [static]

(1043) OCTRL19 (#0x413, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_OCTRL20 = 0x414 [static]

(1044) OCTRL20 (#0x414, RWN, init.
: 0x0, min.: 0x0, max.: 0x7)

int REG_MATES_DIOX_MK1_ORP = 0x415 [static]

(1045) ORP (#0x415, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_DIOX_MK1_ORRT = 0x416 [static]

(1046) ORRT (#0x416, RO, init.
: 0x0)

int REG_MATES_DIOX_MK1_CANID = 0x417 [static]

(1047) CANID (#0x417, RWN, init.
: 40, min.: 40, max.: 49)

int REG_MATES_ADC5_MK1_COF01 = 0x600 [static]

(1536) COF01 (#0x600, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF02 = 0x601 [static]

(1537) COF02 (#0x601, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF03 = 0x602 [static]

(1538) COF03 (#0x602, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF04 = 0x603 [static]

(1539) COF04 (#0x603, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF05 = 0x604 [static]

(1540) COF05 (#0x604, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF06 = 0x605 [static]

(1541) COF06 (#0x605, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF07 = 0x606 [static]

(1542) COF07 (#0x606, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF08 = 0x607 [static]

(1543) COF08 (#0x607, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF09 = 0x608 [static]

(1544) COF09 (#0x608, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF10 = 0x609 [static]

(1545) COF10 (#0x609, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF11 = 0x60A [static]

(1546) COF11 (#0x60A, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF12 = 0x60B [static]

(1547) COF12 (#0x60B, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF13 = 0x60C [static]

(1548) COF13 (#0x60C, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF14 = 0x60D [static]

(1549) COF14 (#0x60D, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF15 = 0x60E [static]

(1550) COF15 (#0x60E, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF16 = 0x60F [static]

(1551) COF16 (#0x60F, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF17 = 0x610 [static]

(1552) COF17 (#0x610, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF18 = 0x611 [static]

(1553) COF18 (#0x611, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF19 = 0x612 [static]

(1554) COF19 (#0x612, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF20 = 0x613 [static]

(1555) COF20 (#0x613, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF21 = 0x614 [static]

(1556) COF21 (#0x614, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF22 = 0x615 [static]

(1557) COF22 (#0x615, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF23 = 0x616 [static]

(1558) COF23 (#0x616, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF24 = 0x617 [static]

(1559) COF24 (#0x617, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF25 = 0x618 [static]

(1560) COF25 (#0x618, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF26 = 0x619 [static]

(1561) COF26 (#0x619, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF27 = 0x61A [static]

(1562) COF27 (#0x61A, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF28 = 0x61B [static]

(1563) COF28 (#0x61B, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF29 = 0x61C [static]

(1564) COF29 (#0x61C, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF30 = 0x61D [static]

(1565) COF30 (#0x61D, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF31 = 0x61E [static]

(1566) COF31 (#0x61E, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF32 = 0x61F [static]

(1567) COF32 (#0x61F, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF33 = 0x620 [static]

(1568) COF33 (#0x620, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF34 = 0x621 [static]

(1569) COF34 (#0x621, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF35 = 0x622 [static]

(1570) COF35 (#0x622, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF36 = 0x623 [static]

(1571) COF36 (#0x623, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF37 = 0x624 [static]

(1572) COF37 (#0x624, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF38 = 0x625 [static]

(1573) COF38 (#0x625, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF39 = 0x626 [static]

(1574) COF39 (#0x626, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_COF40 = 0x627 [static]

(1575) COF40 (#0x627, RWN, init.
: 0.0, min.: -0.1, max.: 0.1)

int REG_MATES_ADC5_MK1_CGA01 = 0x628 [static]

(1576) CGA01 (#0x628, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA02 = 0x629 [static]

(1577) CGA02 (#0x629, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA03 = 0x62A [static]

(1578) CGA03 (#0x62A, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA04 = 0x62B [static]

(1579) CGA04 (#0x62B, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA05 = 0x62C [static]

(1580) CGA05 (#0x62C, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA06 = 0x62D [static]

(1581) CGA06 (#0x62D, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA07 = 0x62E [static]

(1582) CGA07 (#0x62E, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA08 = 0x62F [static]

(1583) CGA08 (#0x62F, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA09 = 0x630 [static]

(1584) CGA09 (#0x630, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA10 = 0x631 [static]

(1585) CGA10 (#0x631, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA11 = 0x632 [static]

(1586) CGA11 (#0x632, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA12 = 0x633 [static]

(1587) CGA12 (#0x633, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA13 = 0x634 [static]

(1588) CGA13 (#0x634, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA14 = 0x635 [static]

(1589) CGA14 (#0x635, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA15 = 0x636 [static]

(1590) CGA15 (#0x636, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA16 = 0x637 [static]

(1591) CGA16 (#0x637, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA17 = 0x638 [static]

(1592) CGA17 (#0x638, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA18 = 0x639 [static]

(1593) CGA18 (#0x639, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA19 = 0x63A [static]

(1594) CGA19 (#0x63A, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA20 = 0x63B [static]

(1595) CGA20 (#0x63B, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA21 = 0x63C [static]

(1596) CGA21 (#0x63C, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA22 = 0x63D [static]

(1597) CGA22 (#0x63D, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA23 = 0x63E [static]

(1598) CGA23 (#0x63E, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA24 = 0x63F [static]

(1599) CGA24 (#0x63F, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA25 = 0x640 [static]

(1600) CGA25 (#0x640, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA26 = 0x641 [static]

(1601) CGA26 (#0x641, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA27 = 0x642 [static]

(1602) CGA27 (#0x642, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA28 = 0x643 [static]

(1603) CGA28 (#0x643, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA29 = 0x644 [static]

(1604) CGA29 (#0x644, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA30 = 0x645 [static]

(1605) CGA30 (#0x645, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA31 = 0x646 [static]

(1606) CGA31 (#0x646, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA32 = 0x647 [static]

(1607) CGA32 (#0x647, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA33 = 0x648 [static]

(1608) CGA33 (#0x648, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA34 = 0x649 [static]

(1609) CGA34 (#0x649, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA35 = 0x64A [static]

(1610) CGA35 (#0x64A, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA36 = 0x64B [static]

(1611) CGA36 (#0x64B, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA37 = 0x64C [static]

(1612) CGA37 (#0x64C, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA38 = 0x64D [static]

(1613) CGA38 (#0x64D, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA39 = 0x64E [static]

(1614) CGA39 (#0x64E, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_CGA40 = 0x64F [static]

(1615) CGA40 (#0x64F, RWN, init.
: 1.0, min.: 0.95, max.: 1.05)

int REG_MATES_ADC5_MK1_LCT = 0x650 [static]

(1616) LCT (#0x650, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_ADC5_MK1_CANID = 0x651 [static]

(1617) CANID (#0x651, RWN, init.
: 60, min.: 60, max.: 69)

int REG_MATES_UCC_MK1_ODEF = 0x300 [static]

(768) ODEF (#0x300, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFF)

int REG_MATES_UCC_MK1_ODIS = 0x301 [static]

(769) ODIS (#0x301, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFF)

int REG_MATES_UCC_MK1_PRD01 = 0x302 [static]

(770) PRD01 (#0x302, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD02 = 0x303 [static]

(771) PRD02 (#0x303, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD03 = 0x304 [static]

(772) PRD03 (#0x304, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD04 = 0x305 [static]

(773) PRD04 (#0x305, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD05 = 0x306 [static]

(774) PRD05 (#0x306, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD06 = 0x307 [static]

(775) PRD06 (#0x307, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD07 = 0x308 [static]

(776) PRD07 (#0x308, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD08 = 0x309 [static]

(777) PRD08 (#0x309, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD09 = 0x30A [static]

(778) PRD09 (#0x30A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD10 = 0x30B [static]

(779) PRD10 (#0x30B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD11 = 0x30C [static]

(780) PRD11 (#0x30C, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD12 = 0x30D [static]

(781) PRD12 (#0x30D, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD13 = 0x30E [static]

(782) PRD13 (#0x30E, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_PRD14 = 0x30F [static]

(783) PRD14 (#0x30F, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE01 = 0x310 [static]

(784) TRISE01 (#0x310, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE02 = 0x311 [static]

(785) TRISE02 (#0x311, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE03 = 0x312 [static]

(786) TRISE03 (#0x312, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE04 = 0x313 [static]

(787) TRISE04 (#0x313, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE05 = 0x314 [static]

(788) TRISE05 (#0x314, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE06 = 0x315 [static]

(789) TRISE06 (#0x315, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE07 = 0x316 [static]

(790) TRISE07 (#0x316, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE08 = 0x317 [static]

(791) TRISE08 (#0x317, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE09 = 0x318 [static]

(792) TRISE09 (#0x318, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE10 = 0x319 [static]

(793) TRISE10 (#0x319, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE11 = 0x31A [static]

(794) TRISE11 (#0x31A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE12 = 0x31B [static]

(795) TRISE12 (#0x31B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE13 = 0x31C [static]

(796) TRISE13 (#0x31C, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TRISE14 = 0x31D [static]

(797) TRISE14 (#0x31D, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL01 = 0x31E [static]

(798) TFALL01 (#0x31E, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL02 = 0x31F [static]

(799) TFALL02 (#0x31F, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL03 = 0x320 [static]

(800) TFALL03 (#0x320, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL04 = 0x321 [static]

(801) TFALL04 (#0x321, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL05 = 0x322 [static]

(802) TFALL05 (#0x322, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL06 = 0x323 [static]

(803) TFALL06 (#0x323, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL07 = 0x324 [static]

(804) TFALL07 (#0x324, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL08 = 0x325 [static]

(805) TFALL08 (#0x325, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL09 = 0x326 [static]

(806) TFALL09 (#0x326, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL10 = 0x327 [static]

(807) TFALL10 (#0x327, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL11 = 0x328 [static]

(808) TFALL11 (#0x328, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL12 = 0x329 [static]

(809) TFALL12 (#0x329, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL13 = 0x32A [static]

(810) TFALL13 (#0x32A, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_TFALL14 = 0x32B [static]

(811) TFALL14 (#0x32B, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ORP = 0x32C [static]

(812) ORP (#0x32C, RWN, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ORRT = 0x32D [static]

(813) ORRT (#0x32D, RO, init.
: 0x0)

int REG_MATES_UCC_MK1_CANID = 0x32E [static]

(814) CANID (#0x32E, RWN, init.
: 30, min.: 30, max.: 39)

int REG_MATES_UCC_MK1_PWMD = 0x32F [static]

(815) PWMD (#0x32F, RW, init.
: 0.0, min.: 0.0, max.: 1.0)

int REG_MATES_UCC_MK1_ALCDB01 = 0x330 [static]

(816) ALCDB01 (#0x330, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB02 = 0x331 [static]

(817) ALCDB02 (#0x331, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB03 = 0x332 [static]

(818) ALCDB03 (#0x332, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB04 = 0x333 [static]

(819) ALCDB04 (#0x333, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB05 = 0x334 [static]

(820) ALCDB05 (#0x334, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB06 = 0x335 [static]

(821) ALCDB06 (#0x335, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB07 = 0x336 [static]

(822) ALCDB07 (#0x336, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

int REG_MATES_UCC_MK1_ALCDB08 = 0x337 [static]

(823) ALCDB08 (#0x337, RW, init.
: 0x0, min.: 0x0, max.: 0xFFFFFFFF)

The documentation for this class was generated from the following file:

- regs.py

A.16 Mates Class Reference

MATES system wrapper class.

Public Member Functions

- Mates (string fileName, int addr=0)

  Create the MATES instance.
- bool DiscoverSingleNode (int addr)

  Discover single MATES node using its CAN address.
- bool DiscoverSingleNode (NodeId addr)

  Discover single MATES node using its CAN address.
- Node NewNode (NodeId addr)

  Discover and create a new MATES-DIOX-MK1 node.
- DioxNode NewDioxNode (NodeId addr)

  Discover and create a new MATES-DIOX-MK1 node.
- Dio3Node NewDio3Node (NodeId addr)

  Discover and create a new MATES-DIO3-MK1 node.
- UccNode NewUccNode (NodeId addr)

  Discover and create a new MATES-UCC-MK1 node.
- Adc5Node NewAdc5Node (NodeId addr)

  Discover and create a new MATES-ADC5-MK1 node.
- Dac5Node NewDac5Node (NodeId addr)

  Discover and create a new MATES-DAC5-MK1 node.
- int GetRegisterI (int addr, RegisterNumber reg)

  Get regular (integral) MATES register.
- float GetRegisterF (int addr, RegisterNumber reg)

  Get floating point MATES register.
- T GetRegister< T > (int addr, RegisterNumber reg)

  Get MATES register.
- void SetRegister< T > (int addr, RegisterNumber reg, T value)

  Set MATES register.
- void SetRegisterI (int addr, RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (int addr, RegisterNumber reg, float val)

  Set floating point MATES register.
- int GetAdcRaw (int addr, int channel)

  Get the raw value of the ADC.
- List< Node > DiscoverNodes (int[] addr)

  Discover MATES nodes.
- List< Node > DiscoverNodes (NodeId[] addr)

  Discover MATES nodes.
- async Task< List< Node > > DiscoverNodesAsync (NodeId[] addr)

  Discover MATES nodes, asynchronous version.

- string GetLastError ()

  Returns the description of lastly occurred error.
- string GetError ()

  Get the accumulated error description.
- int LampTest (int addr)

  Execute the lamp test.
- string NodeInfo (int addr)

  Get node information as string.
- void AlcdWrite (int addr, int row, int col, string text)

  Write text onto the alphanumeric LCD display.

Static Public Member Functions

- static Mates FromBuffer (string confString, int addr=0)

  Create the MATES instance.
- static T DllVersion< T > ()

  Get mates.dll file version.

Properties

- DinStateChangeHandler OnDinStateChange

  Event raised when any digital input in the MATES system changes.
- DoutStateChangeHandler OnDoutStateChange

  Event raised when any digital output in the MATES system changes.
- AdcStateChangeHandler OnAdcStateChange

  Event raised when any analog input in the MATES system changes value.

A.16.1 Detailed Description

MATES system wrapper class.

A.16.2 Constructor & Destructor Documentation

Mates (
string *fileName*,
int *addr* = *0*)

Create the MATES instance.

Parameters

fileName	Defines the name of the monitor configuration file(∗.mon).
addr	The MATES device communication address to use

A.16.3 Member Function Documentation

static Mates FromBuffer (
string *confString*,
int *addr* = *0*) `[static]`

Create the MATES instance.

Parameters

confString	The monitor XML configuration string.
addr	The MATES device communication address to use

static T DllVersion< T > ( ) [static]

Get mates.dll file version.

Template Parameters

T	The return value type, either string or Tuple<int, int, int, int>

Returns The version.

Exceptions

MatesException	Thrown when operation fails.

bool DiscoverSingleNode (
int *addr*)

Discover single MATES node using its CAN address.

Parameters

addr	The device CAN address, see NodeId.

Returns True when the device was found or false otherwise.

bool DiscoverSingleNode (
NodeId *addr*)

Discover single MATES node using its CAN address.

Parameters

addr	The device CAN address.

Returns True when the device was found or false otherwise.

Node NewNode (
NodeId *addr*)

Discover and create a new MATES-DIOX-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

DioxNode NewDioxNode (
NodeId *addr*)

Discover and create a new MATES-DIOX-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

Dio3Node NewDio3Node (
NodeId *addr*)

Discover and create a new MATES-DIO3-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

UccNode NewUccNode (
NodeId *addr*)

Discover and create a new MATES-UCC-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

Adc5Node NewAdc5Node (
NodeId *addr*)

Discover and create a new MATES-ADC5-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

Dac5Node NewDac5Node (
NodeId *addr*)

Discover and create a new MATES-DAC5-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

int GetRegisterI (
int *addr*,
RegisterNumber *reg*)

Get regular (integral) MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

Exceptions

MatesException	Thrown when operation fails.

float GetRegisterF (
int *addr*,
RegisterNumber *reg*)

Get floating point MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

Exceptions

MatesException	Thrown when operation fails.

T GetRegister< T > (
int *addr*,
RegisterNumber *reg*)

Get MATES register.

Template Parameters

T	The register type, float or int.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

Exceptions

MatesException	Thrown when operation fails.

Type Constraints T : *struct* :

void SetRegister< T > (
int *addr*,
RegisterNumber *reg*,
T *value*)

Set MATES register.

Template Parameters

T	The register type, float or int.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
value	The register value.

Exceptions

MatesException	Thrown when operation fails.

Type Constraints T : *struct* :

void SetRegisterI (
int *addr*,
RegisterNumber *reg*,
int *val*)

Set regular (integral) MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

Exceptions

MatesException	Thrown when operation fails.

void SetRegisterF (
int *addr*,
RegisterNumber *reg*,
float *val*)

Set floating point MATES register.

Parameters

addr	The node CAN address.

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

Exceptions

MatesException	Thrown when operation fails.

int GetAdcRaw (
int *addr*,
int *channel*)

Get the raw value of the ADC.

Parameters

addr	The node address.
channel	ADC channel number [0, 39].

Returns The raw ADC value.

Exceptions

MatesException	Thrown when operation fails.

List<Node> DiscoverNodes (
int[] *addr*)

Discover MATES nodes.

Parameters

addr	List of nodes addresses to find.

Returns Returns the list of nodes found or empty list on error.

List<Node> DiscoverNodes (
NodeId[] *addr*)

Discover MATES nodes.

Parameters

addr	List of nodes addresses to find.

Returns Returns the list of nodes found or empty list on error.

async Task<List<Node> > DiscoverNodesAsync (
NodeId[] *addr*)

Discover MATES nodes, asynchronous version.

Parameters

addr	List of nodes addresses to find.

Returns Returns the list of nodes found or empty list on error.

string GetLastError ( )

Returns the description of lastly occurred error.

Returns The error description.

string GetError ( )

Get the accumulated error description.

Returns The buffered errors or empty string.

int LampTest (
int *addr*)

Execute the lamp test.

Parameters

addr	The node address.

Exceptions

MatesException	Thrown when operation fails.

string NodeInfo (
int *addr*)

Get node information as string.

Parameters

addr	The node address.

Returns The node information in a textual form.

Exceptions

MatesException	Thrown when operation fails.

void AlcdWrite (
int *addr*,
int *row*,
int *col*,
string *text*)

Write text onto the alphanumeric LCD display.

Parameters

addr	The node address.
row	The LCD row number (0 - based).
col	The LCD column number (0 - based).
text	The text to write at the given position.

Exceptions

MatesException	Thrown when operation fails.

A.16.4 Property Documentation

DinStateChangeHandler OnDinStateChange `[add], [remove]`

Event raised when any digital input in the MATES system changes.

The state change sampling period is roughly 50 ms and it increases with the number of nodes in the system.

The first subscriber will receive states of all inputs even if they are not changing (this action is performed only once after subscribing). This is done for the subscriber to be able to determine the initial state after device connection. The same procedure is repeated every time the subscribers list becomes empty and then re-populated.

DoutStateChangeHandler OnDoutStateChange `[add], [remove]`

Event raised when any digital output in the MATES system changes.

The state change sampling period is roughly 50 ms and it increases with the number of nodes in the system.

The first subscriber will receive states of all outputs even if they are not changing (this action is performed only once after subscribing). This is done for the subscriber to be able to determine the initial state after device connection. The same procedure is repeated every time the subscribers list becomes empty and then re-populated.

AdcStateChangeHandler OnAdcStateChange `[add], [remove]`

Event raised when any analog input in the MATES system changes value.

The value change sampling period is roughly 50 ms and it increases with the number of nodes in the system.

The first subscriber will receive values of all inputs even if they are not changing (this action is performed only once after subscribing). This is done for the subscriber to be able to determine the initial state after device connection. The same procedure is repeated every time the subscribers list becomes empty and then re-populated.

The documentation for this class was generated from the following file:

- Mates.cs

A.17 MatesException Class Reference

Represents exception used by MATES to communicate runtime errors.

A.17.1 Detailed Description

Represents exception used by MATES to communicate runtime errors.

The documentation for this class was generated from the following file:

- MatesExceptions.cs

A.18 MatesRangeError Class Reference

Represents exception used by MATES to signal range errors.

A.18.1 Detailed Description

Represents exception used by MATES to signal range errors.

The documentation for this class was generated from the following file:

- MatesExceptions.cs

A.19 MatesConfigError Class Reference

Represents exception used by MATES to communicate configuration errors.

A.19.1 Detailed Description

Represents exception used by MATES to communicate configuration errors.

The documentation for this class was generated from the following file:

- MatesExceptions.cs

A.20 Viresco.Mates Namespace Reference

MATES system main classes.

Namespaces

- namespace Registers

  The MATES registers definitions.

Data Structures

- class Adc5Node

  Represents single MATES-ADC5-MK1 node.
- class AdcNode

  Represents a common ADC module.
- class Dac5Node

  Represents single MATES-DAC5-MK1 node.
- class DacNode

  Represents a common DAC module.
- class Dio3Node

  Represents single MATES-DIOX-MK1 node.
- class DioNode

  Represents a common digital IO module.
- class DioxNode

  Represents single MATES-DIOX-MK1 node.
- interface IAdcNode

  Represents a common ADC module.
- interface IDacNode

  Represents a common DAC module.
- interface IDioAsserts

  Asserts suitable for a DIO node.
- interface INodeAsserts

  Asserts available for all types of nodes.
- class Mates

  MATES system wrapper class.
- class MatesConfigError

  Represents exception used by MATES to communicate configuration errors.
- class MatesException

  Represents exception used by MATES to communicate runtime errors.
- class MatesRangeError

  Represents exception used by MATES to signal range errors.
- class Node

  Represents basic node with features common to all nodes.
- class Register

  Represents abstract Mates register.

- class UccNode

  Represents single MATES-UCC-MK1 node.

Enumerations

A.20.1 Detailed Description

MATES system main classes.

A.20.2 Enumeration Type Documentation

enum NodeId `[strong]`

Node address enumeration.

Examples: mates_test_01.cs, mates_test_02.cs, mates_test_03.cs, mates_test_04.cs, mates_test_05.cs, mates_test_06.cs, mates_test_07.cs, mates_test_08.cs, mates_test_09.cs, mates_test_10.cs, mates_test_11.cs, mates_test_13.cs, mates_test_14.cs, and mates_test_15.cs.

enum OutputChannelNumber `[strong]`

MATES output channels enumeration.

Enumerator *MATES_OUT01* : (0) Output #1 - front panel designation: OUT01.

MATES_OUT02 : (1) Output #2 - front panel designation: OUT02.

MATES_OUT03 : (2) Output #3 - front panel designation: OUT03.

MATES_OUT04 : (3) Output #4 - front panel designation: OUT04.

MATES_OUT05 : (4) Output #5 - front panel designation: OUT05.

MATES_OUT06 : (5) Output #6 - front panel designation: OUT06.

MATES_OUT07 : (6) Output #7 - front panel designation: OUT07.

MATES_OUT08 : (7) Output #8 - front panel designation: OUT08.

MATES_OUT09 : (8) Output #9 - front panel designation: OUT09.

MATES_OUT10 : (9) Output #10 - front panel designation: OUT10.

MATES_OUT11 : (10) Output #11 - front panel designation: OUT11.

MATES_OUT12 : (11) Output #12 - front panel designation: OUT12.

MATES_OUT13 : (12) Output #13 - front panel designation: OUT13.

MATES_OUT14 : (13) Output #14 - front panel designation: OUT14.

MATES_OUT15 : (14) Output #15 - front panel designation: OUT15.

MATES_OUT16 : (15) Output #16 - front panel designation: OUT16.

MATES_OUT17 : (16) Output #17 - front panel designation: OUT17.

MATES_OUT18 : (17) Output #18 - front panel designation: OUT18.

MATES_OUT19 : (18) Output #19 - front panel designation: OUT19.

MATES_OUT20 : (19) Output #20 - front panel designation: OUT20.

MATES_OUT21 : (20) Output #21 - front panel designation: OUT21.

MATES_OUT22 : (21) Output #22 - front panel designation: OUT22.

MATES_OUT23 : (22) Output #23 - front panel designation: OUT23.

MATES_OUT24 : (23) Output #24 - front panel designation: OUT24.

MATES_OUT25 : (24) Output #25 - front panel designation: OUT25.

MATES_OUT26 : (25) Output #26 - front panel designation: OUT26.

MATES_OUT27 : (26) Output #27 - front panel designation: OUT27.

MATES_OUT28 : (27) Output #28 - front panel designation: OUT28.

MATES_OUT29 : (28) Output #29 - front panel designation: OUT29.

MATES_OUT30 : (29) Output #30 - front panel designation: OUT30.

MATES_OUT31 : (30) Output #31 - front panel designation: OUT31.

MATES_OUT32 : (31) Output #32 - front panel designation: OUT32.

MATES_OUT33 : (32) Output #33 - front panel designation: OUT33.

MATES_OUT34 : (33) Output #34 - front panel designation: OUT34.

MATES_OUT35 : (34) Output #35 - front panel designation: OUT35.

MATES_OUT36 : (35) Output #36 - front panel designation: OUT36.

MATES_OUT37 : (36) Output #37 - front panel designation: OUT37.

MATES_OUT38 : (37) Output #38 - front panel designation: OUT38.

MATES_OUT39 : (38) Output #39 - front panel designation: OUT39.

MATES_OUT40 : (39) Output #40 - front panel designation: OUT40.

enum InputChannelNumber `[strong]`

Enumerator *MATES_IN01* : (0) Input #1 - front panel designation: IN01.

MATES_IN02 : (1) Input #2 - front panel designation: IN02.

MATES_IN03 : (2) Input #3 - front panel designation: IN03.

MATES_IN04 : (3) Input #4 - front panel designation: IN04.

MATES_IN05 : (4) Input #5 - front panel designation: IN05.

MATES_IN06 : (5) Input #6 - front panel designation: IN06.

MATES_IN07 : (6) Input #7 - front panel designation: IN07.

MATES_IN08 : (7) Input #8 - front panel designation: IN08.

MATES_IN09 : (8) Input #9 - front panel designation: IN09.

MATES_IN10 : (9) Input #10 - front panel designation: IN10.

MATES_IN11 : (10) Input #11 - front panel designation: IN11.

MATES_IN12 : (11) Input #12 - front panel designation: IN12.

MATES_IN13 : (12) Input #13 - front panel designation: IN13.

MATES_IN14 : (13) Input #14 - front panel designation: IN14.

MATES_IN15 : (14) Input #15 - front panel designation: IN15.

MATES_IN16 : (15) Input #16 - front panel designation: IN16.

MATES_IN17 : (16) Input #17 - front panel designation: IN17.

MATES_IN18 : (17) Input #18 - front panel designation: IN18.

MATES_IN19 : (18) Input #19 - front panel designation: IN19.

MATES_IN20 : (19) Input #20 - front panel designation: IN20.

MATES_IN21 : (20) Input #21 - front panel designation: IN21.

MATES_IN22 : (21) Input #22 - front panel designation: IN22.

MATES_IN23 : (22) Input #23 - front panel designation: IN23.

MATES_IN24 : (23) Input #24 - front panel designation: IN24.

MATES_IN25 : (24) Input #25 - front panel designation: IN25.

MATES_IN26 : (25) Input #26 - front panel designation: IN26.

MATES_IN27 : (26) Input #27 - front panel designation: IN27.

MATES_IN28 : (27) Input #28 - front panel designation: IN28.

MATES_IN29 : (28) Input #29 - front panel designation: IN29.

MATES_IN30 : (29) Input #30 - front panel designation: IN30.

MATES_IN31 : (30) Input #31 - front panel designation: IN31.

MATES_IN32 : (31) Input #32 - front panel designation: IN32.

MATES_IN33 : (32) Input #33 - front panel designation: IN33.

MATES_IN34 : (33) Input #34 - front panel designation: IN34.

MATES_IN35 : (34) Input #35 - front panel designation: IN35.

MATES_IN36 : (35) Input #36 - front panel designation: IN36.

MATES_IN37 : (36) Input #37 - front panel designation: IN37.

MATES_IN38 : (37) Input #38 - front panel designation: IN38.

MATES_IN39 : (38) Input #39 - front panel designation: IN39.

MATES_IN40 : (39) Input #40 - front panel designation: IN40.

A.21 Mates Class Reference

MATES system wrapper class.

Public Member Functions

- Mates (string fileName, int addr=0)

  Create the MATES instance.
- bool DiscoverSingleNode (int addr)

  Discover single MATES node using its CAN address.
- bool DiscoverSingleNode (NodeId addr)

  Discover single MATES node using its CAN address.
- Node NewNode (NodeId addr)

  Discover and create a new MATES-DIOX-MK1 node.
- DioxNode NewDioxNode (NodeId addr)

  Discover and create a new MATES-DIOX-MK1 node.

- Dio3Node NewDio3Node (NodeId addr)

  Discover and create a new MATES-DIO3-MK1 node.
- UccNode NewUccNode (NodeId addr)

  Discover and create a new MATES-UCC-MK1 node.
- Adc5Node NewAdc5Node (NodeId addr)

  Discover and create a new MATES-ADC5-MK1 node.
- Dac5Node NewDac5Node (NodeId addr)

  Discover and create a new MATES-DAC5-MK1 node.
- int GetRegisterI (int addr, RegisterNumber reg)

  Get regular (integral) MATES register.
- float GetRegisterF (int addr, RegisterNumber reg)

  Get floating point MATES register.
- T GetRegister< T > (int addr, RegisterNumber reg)

  Get MATES register.
- void SetRegister< T > (int addr, RegisterNumber reg, T value)

  Set MATES register.
- void SetRegisterI (int addr, RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (int addr, RegisterNumber reg, float val)

  Set floating point MATES register.
- int GetAdcRaw (int addr, int channel)

  Get the raw value of the ADC.
- List< Node > DiscoverNodes (int[] addr)

  Discover MATES nodes.
- List< Node > DiscoverNodes (NodeId[] addr)

  Discover MATES nodes.
- async Task< List< Node > > DiscoverNodesAsync (NodeId[] addr)

  Discover MATES nodes, asynchronous version.
- string GetLastError ()

  Returns the description of lastly occurred error.
- string GetError ()

  Get the accumulated error description.
- int LampTest (int addr)

  Execute the lamp test.
- string NodeInfo (int addr)

  Get node information as string.
- void AlcdWrite (int addr, int row, int col, string text)

  Write text onto the alphanumeric LCD display.

Static Public Member Functions

- static Mates FromBuffer (string confString, int addr=0)

  Create the MATES instance.
- static T DllVersion< T > ()

  Get mates.dll file version.

Properties

- DinStateChangeHandler OnDinStateChange

  Event raised when any digital input in the MATES system changes.
- DoutStateChangeHandler OnDoutStateChange

  Event raised when any digital output in the MATES system changes.
- AdcStateChangeHandler OnAdcStateChange

  Event raised when any analog input in the MATES system changes value.

A.21.1 Detailed Description

MATES system wrapper class.

A.21.2 Constructor & Destructor Documentation

Mates (
string *fileName*,
int *addr* = *0*)

Create the MATES instance.

Parameters

fileName	Defines the name of the monitor configuration file(∗.mon).
addr	The MATES device communication address to use

A.21.3 Member Function Documentation

static Mates FromBuffer (
string *confString*,
int *addr* = *0*) `[static]`

Create the MATES instance.

Parameters

confString	The monitor XML configuration string.
addr	The MATES device communication address to use

static T DllVersion< T > ( ) `[static]`

Get mates.dll file version.

Template Parameters

T	The return value type, either string or Tuple<int, int, int, int>

Returns The version.

Exceptions

MatesException	Thrown when operation fails.

bool DiscoverSingleNode (
int *addr*)

Discover single MATES node using its CAN address.

Parameters

addr	The device CAN address, see NodeId.

Returns True when the device was found or false otherwise.

bool DiscoverSingleNode (
NodeId *addr*)

Discover single MATES node using its CAN address.

Parameters

addr	The device CAN address.

Returns True when the device was found or false otherwise.

Node NewNode (
NodeId *addr*)

Discover and create a new MATES-DIOX-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

DioxNode NewDioxNode (
NodeId *addr*)

Discover and create a new MATES-DIOX-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

Dio3Node NewDio3Node (
NodeId *addr*)

Discover and create a new MATES-DIO3-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

UccNode NewUccNode (
NodeId *addr*)

Discover and create a new MATES-UCC-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

Adc5Node NewAdc5Node (
NodeId *addr*)

Discover and create a new MATES-ADC5-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

Dac5Node NewDac5Node (
NodeId *addr*)

Discover and create a new MATES-DAC5-MK1 node.

Parameters

addr	The node address to use.

Returns The created node.

Exceptions

MatesException	Thrown when the device cannot be found in the system.

int GetRegisterI (
int *addr*,
RegisterNumber *reg*)

Get regular (integral) MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

Exceptions

MatesException	Thrown when operation fails.

float GetRegisterF (
int *addr*,
RegisterNumber *reg*)

Get floating point MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

Exceptions

MatesException	Thrown when operation fails.

T GetRegister< T > (
int *addr*,
RegisterNumber *reg*)

Get MATES register.

Template Parameters

T	The register type, float or int.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

Exceptions

MatesException	Thrown when operation fails.

Type Constraints *T* : *struct* :

void SetRegister< T > (
int *addr*,
RegisterNumber *reg*,
T *value*)

Set MATES register.

Template Parameters

T	The register type, float or int.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
value	The register value.

Exceptions

MatesException	Thrown when operation fails.

Type Constraints *T* : *struct* :

void SetRegisterI (
int *addr*,
RegisterNumber *reg*,
int *val*)

Set regular (integral) MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

Exceptions

MatesException	Thrown when operation fails.

void SetRegisterF (
int *addr*,
RegisterNumber *reg*,
float *val*)

Set floating point MATES register.

Parameters

addr	The node CAN address.
reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

Exceptions

MatesException	Thrown when operation fails.

int GetAdcRaw (
int *addr*,
int *channel*)

Get the raw value of the ADC.

Parameters

addr	The node address.
channel	ADC channel number [0, 39].

Returns The raw ADC value.

Exceptions

MatesException	Thrown when operation fails.

List<Node> DiscoverNodes (
int[] *addr*)

Discover MATES nodes.

Parameters

addr	List of nodes addresses to find.

Returns Returns the list of nodes found or empty list on error.

List<Node> DiscoverNodes (
NodeId[] *addr*)

Discover MATES nodes.

Parameters

addr	List of nodes addresses to find.

Returns Returns the list of nodes found or empty list on error.

async Task<List<Node> > DiscoverNodesAsync (
NodeId[] *addr*)

Discover MATES nodes, asynchronous version.

Parameters

addr	List of nodes addresses to find.

Returns Returns the list of nodes found or empty list on error.

string GetLastError ( )

Returns the description of lastly occurred error.

Returns The error description.

string GetError ( )

Get the accumulated error description.

Returns The buffered errors or empty string.

int LampTest (
int *addr*)

Execute the lamp test.

Parameters

addr	The node address.

Exceptions

MatesException	Thrown when operation fails.

string NodeInfo (
int *addr*)

Get node information as string.

Parameters

addr	The node address.

Returns The node information in a textual form.

Exceptions

MatesException	Thrown when operation fails.

void AlcdWrite (
int *addr*,
int *row*,
int *col*,
string *text*)

Write text onto the alphanumeric LCD display.

Parameters

addr	The node address.
row	The LCD row number (0 - based).
col	The LCD column number (0 - based).
text	The text to write at the given position.

Exceptions

MatesException	Thrown when operation fails.

A.21.4 Property Documentation

DinStateChangeHandler OnDinStateChange `[add]`, `[remove]`

Event raised when any digital input in the MATES system changes.

The state change sampling period is roughly 50 ms and it increases with the number of nodes in the system.

The first subscriber will receive states of all inputs even if they are not changing (this action is performed only once after subscribing). This is done for the subscriber to be able to determine the initial state after device connection. The same procedure is repeated every time the subscribers list becomes empty and then re-populated.

DoutStateChangeHandler OnDoutStateChange `[add]`, `[remove]`

Event raised when any digital output in the MATES system changes.

The state change sampling period is roughly 50 ms and it increases with the number of nodes in the system.

The first subscriber will receive states of all outputs even if they are not changing (this action is performed only once after subscribing). This is done for the subscriber to be able to determine the initial state after device connection. The same procedure is repeated every time the subscribers list becomes empty and then re-populated.

AdcStateChangeHandler OnAdcStateChange `[add], [remove]`

Event raised when any analog input in the MATES system changes value.

The value change sampling period is roughly 50 ms and it increases with the number of nodes in the system.

The first subscriber will receive values of all inputs even if they are not changing (this action is performed only once after subscribing). This is done for the subscriber to be able to determine the initial state after device connection. The same procedure is repeated every time the subscribers list becomes empty and then re-populated.

The documentation for this class was generated from the following file:

- Mates.cs

A.22 INodeAsserts Interface Reference

Asserts available for all types of nodes.

Public Member Functions

- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.

A.22.1 Detailed Description

Asserts available for all types of nodes.

A.22.2 Member Function Documentation

void RegisterEquals (
RegisterNumber *register*,
int *value*)

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implemented in Node.

void RegisterEquals (
RegisterNumber *register*,
float *value*)

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implemented in Node.

The documentation for this interface was generated from the following file:

- Node.cs

A.23 Node Class Reference

Represents basic node with features common to all nodes.

Public Member Functions

- bool IsCanBridge ()

  Check if this node acts as a CAN bridge (it is connected directly to PC).
- int GetStatusRegister ()

  Get the value of the status register.
- int GetRegisterI (RegisterNumber reg)

  Get regular (integral) MATES register.
- float GetRegisterF (RegisterNumber reg)

  Get floating point MATES register.
- void SetRegisterI (RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (RegisterNumber reg, float val)

  Set floating point MATES register.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.
- void LampTest ()

  Execute lamp test.
- string NodeInfo ()

  Get node information as string.

Data Fields

- RegList Registers

  The registers that are associated with this node. Populated by the node constructor.

Properties

- int SerialNumber `[get]`

  Holds the device serial number.
- string NodeIdName `[get]`

  Get string representation of this node Id (NodeId).
- string NodeName `[get]`

Get node name of this node.

A.23.1 Detailed Description

Represents basic node with features common to all nodes.

A.23.2 Member Function Documentation

bool IsCanBridge ( )

Check if this node acts as a CAN bridge (it is connected directly to PC).

Returns True if CAN bridge, otherwise false.

int GetStatusRegister ( )

Get the value of the status register.

Returns The REG_COMMON_SR register value.

int GetRegisterI (
RegisterNumber *reg*)

Get regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

float GetRegisterF (
RegisterNumber *reg*)

Get floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

void SetRegisterI (
RegisterNumber *reg*,
int *val*)

Set regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void SetRegisterF (
RegisterNumber *reg*,
float *val*)

Set floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void RegisterEquals (
RegisterNumber *register*,
int *value*)

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void RegisterEquals (
RegisterNumber *register*,
float *value*)

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void LampTest ( )

Execute lamp test.

Example usage (mates_test_11.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_11
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Locate node by blinking its LEDs.
               dio.LampTest();
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
```

```
        }
      }
   }
}
```

string NodeInfo ( )

Get node information as string.

Returns The node information.

Example usage (mates_test_13.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_13
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode dio = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
               // Get node info and print to console.
               string info = dio.NodeInfo();
               Console.Write(info);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

A.23.3 Field Documentation

RegList Registers

The registers that are associated with this node. Populated by the node constructor.

A.23.4 Property Documentation

int SerialNumber `[get]`

Holds the device serial number.

string NodeIdName `[get]`

Get string representation of this node Id (NodeId).

string NodeName `[get]`

Get node name of this node.

The documentation for this class was generated from the following files:

- Node.cs
- MatesTypes.cs

A.24 IDioAsserts Interface Reference

Asserts suitable for a DIO node.

Public Member Functions

- void InputSet (int channel)

  Asserts that the specified input channel is in high state.
- void InputEquals (int channel, bool state)

  Asserts that the specified input channel is in the given state.
- void OutputSet (int channel)

  Asserts that the specified output channel is in high state.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.

A.24.1 Detailed Description

Asserts suitable for a DIO node.

A.24.2 Member Function Documentation

void InputSet (
int *channel*)

Asserts that the specified input channel is in high state.

Parameters

channel	The input channel number [0, 19].

Implemented in DioNode.

void InputEquals (
int *channel,*
bool *state*)

Asserts that the specified input channel is in the given state.

Parameters

channel	The input channel number [0, 19].
state	The expected state.

Implemented in DioNode.

void OutputSet (
int *channel*)

Asserts that the specified output channel is in high state.

Parameters

channel	The output channel number [0, 19].

Implemented in DioNode.

void RegisterEquals (
RegisterNumber *register*,
int *value*) [inherited]

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implemented in Node.

void RegisterEquals (
RegisterNumber *register*,
float *value*) [inherited]

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implemented in Node.

The documentation for this interface was generated from the following file:

- DioNode.cs

A.25 Dio3Node Class Reference

Represents single MATES-DIOX-MK1 node.

Public Member Functions

- void SetDout (int channel, bool value)

  Set value of an individual digital output.
- void SetDoutAll (int value)

  Set value of all digital outputs.
- bool GetDout (int channel)

  Get value of an individual digital output.
- int GetDoutAll ()

  Get value of all digital outputs.
- bool GetDin (int channel)

  Get value of an individual digital input.
- int GetDinAll ()

  Get value of all digital inputs.
- bool IsCanBridge ()

  Check if this node acts as a CAN bridge (it is connected directly to PC).
- int GetStatusRegister ()

  Get the value of the status register.
- int GetRegisterI (RegisterNumber reg)

  Get regular (integral) MATES register.

- float GetRegisterF (RegisterNumber reg)

  Get floating point MATES register.
- void SetRegisterI (RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (RegisterNumber reg, float val)

  Set floating point MATES register.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.
- void LampTest ()

  Execute lamp test.
- string NodeInfo ()

  Get node information as string.

Data Fields

- RegList Registers

  The registers that are associated with this node. Populated by the node constructor.

Protected Member Functions

- void InputSet (int channel)

  Asserts that the specified input channel is in high state.
- void OutputSet (int channel)

  Asserts that the specified output channel is in high state.
- void InputEquals (int channel, bool state)

  Asserts that the specified input channel is in the given state.

Properties

- IDioAsserts Assert `[get]`

  Represents collection of UnitTestFramework - compatible asserts defined for this type of node.
- int SerialNumber `[get]`

  Holds the device serial number.
- string NodeIdName `[get]`

  Get string representation of this node Id (NodeId).
- string NodeName `[get]`

  Get node name of this node.

A.25.1 Detailed Description

Represents single MATES-DIOX-MK1 node.

The Dio3Node constructor is not intended to be called directly. Use Mates.New↩ Dio3Node instead.

A.25.2 Member Function Documentation

void SetDout (
int *channel*,
bool *value*) [inherited]

Set value of an individual digital output.

Parameters

channel	The DOUT channel number [0,19].
value	The output value.

Example usage (mates_test_06.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_06
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node node = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               foreach (var val in new bool[]{false, true, false})
               {
                  // Assume loopback is pressent.
                  node.SetDout(0, val);
                  node.Assert.InputEquals(0, val);
               }
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

void SetDoutAll (
int *value*) [inherited]

Set value of all digital outputs.

Parameters

value	The output value, LSB is OUT01.

Example usage (mates_test_05.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_05
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
```

```
                // Assume there is loopback present between outputs and inputs.
                dio.SetDoutAll(0xCAFE);
                Assert.AreEqual(dio.GetDinAll(), 0xCAFE);
            }
            catch
            {
                Console.Write("Cannot discover node.\n");
            }
        }
    }
  }
}
```

bool GetDout (
int *channel*) [inherited]

Get value of an individual digital output.

Parameters

channel	The DOUT channel number [0,19].

Returns The result.

Example usage (mates_test_07.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_07
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Set and verify.
               dio.SetDout(10, true);
               Assert.AreEqual(dio.GetDout(10), true);
               dio.SetDout(10, false);
               Assert.AreEqual(dio.GetDout(10), false);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

int GetDoutAll ( ) [inherited]

Get value of all digital outputs.

Returns The result, LSB is OUT01.

Example usage (mates_test_08.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_08
   {
```

```
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // No more than 20 outputs can be high.
               Assert.IsTrue(dio.GetDoutAll() <= 0xFFFFF);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

bool GetDin (
int *channel*) [inherited]

Get value of an individual digital input.

Parameters

channel	The DIN channel number [0,19].

Returns The result.

Example usage (mates_test_09.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_09
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Enumerate all inputs.
               for (Mates.MatesInput input = Mates.MatesInput.IN01;
                    input <= Mates.MatesInput.IN20;
                    input++)
               {
                  Console.Write("Input {0}: {1}\n", input, dio.GetDin((int)input));
               }
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

int GetDinAll ( ) [inherited]

Get value of all digital inputs.

Returns The result, LSB is IN01.

Example usage (mates_test_10.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_10
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Send 8 of inputs to the outputs.
               int output = dio.GetDoutAll();
               int input = dio.GetDinAll();
               dio.SetDoutAll((output & 0xFFF00) | (input & 0xFF));
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

void InputSet (
int *channel*) `[protected], [inherited]`

Asserts that the specified input channel is in high state.

Parameters

channel	The input channel number [0, 19].

Implements IDioAsserts.

void OutputSet (
int *channel*) `[protected], [inherited]`

Asserts that the specified output channel is in high state.

Parameters

channel	The output channel number [0, 19].

Implements IDioAsserts.

void InputEquals (
int *channel*,
bool *state*) `[protected], [inherited]`

Asserts that the specified input channel is in the given state.

Parameters

channel	The input channel number [0, 19].
state	The expected state.

Implements IDioAsserts.

bool IsCanBridge ( ) [inherited]

Check if this node acts as a CAN bridge (it is connected directly to PC).

Returns True if CAN bridge, otherwise false.

int GetStatusRegister ( ) [inherited]

Get the value of the status register.

Returns The REG_COMMON_SR register value.

int GetRegisterI (
RegisterNumber *reg*) [inherited]

Get regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

float GetRegisterF (
RegisterNumber *reg*) [inherited]

Get floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

void SetRegisterI (
RegisterNumber *reg*,
int *val*) [inherited]

Set regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void SetRegisterF (
RegisterNumber *reg*,
float *val*) [inherited]

Set floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

val	The register value.

void RegisterEquals (
RegisterNumber *register*,
int *value*) [inherited]

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void RegisterEquals (
RegisterNumber *register*,
float *value*) [inherited]

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void LampTest ( ) [inherited]

Execute lamp test.

Example usage (mates_test_11.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_11
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Locate node by blinking its LEDs.
               dio.LampTest();
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

string NodeInfo ( ) [inherited]

Get node information as string.

Returns The node information.

Example usage (mates_test_13.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_13
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode dio = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
               // Get node info and print to console.
               string info = dio.NodeInfo();
               Console.Write(info);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

A.25.3 Field Documentation

RegList Registers [inherited]

The registers that are associated with this node. Populated by the node constructor.

A.25.4 Property Documentation

IDioAsserts Assert [get], [inherited]

Represents collection of UnitTestFramework - compatible asserts defined for this type of node.

int SerialNumber [get], [inherited]

Holds the device serial number.

string NodeIdName [get], [inherited]

Get string representation of this node Id (NodeId).

string NodeName [get], [inherited]

Get node name of this node.

The documentation for this class was generated from the following files:

- Dio3Node.cs

- MatesTypes.cs

A.26 DioxNode Class Reference

Represents single MATES-DIOX-MK1 node.

Public Member Functions

- void SetDout (int channel, bool value)

  Set value of an individual digital output.
- void SetDoutAll (int value)

  Set value of all digital outputs.
- bool GetDout (int channel)

  Get value of an individual digital output.
- int GetDoutAll ()

  Get value of all digital outputs.
- bool GetDin (int channel)

  Get value of an individual digital input.
- int GetDinAll ()

  Get value of all digital inputs.
- bool IsCanBridge ()

  Check if this node acts as a CAN bridge (it is connected directly to PC).
- int GetStatusRegister ()

  Get the value of the status register.
- int GetRegisterI (RegisterNumber reg)

  Get regular (integral) MATES register.
- float GetRegisterF (RegisterNumber reg)

  Get floating point MATES register.
- void SetRegisterI (RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (RegisterNumber reg, float val)

  Set floating point MATES register.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.
- void LampTest ()

  Execute lamp test.
- string NodeInfo ()

  Get node information as string.

Data Fields

- RegList Registers

  The registers that are associated with this node. Populated by the node constructor.

Protected Member Functions

- void InputSet (int channel)

  Asserts that the specified input channel is in high state.
- void OutputSet (int channel)

  Asserts that the specified output channel is in high state.
- void InputEquals (int channel, bool state)

  Asserts that the specified input channel is in the given state.

Properties

- IDioAsserts Assert `[get]`

  Represents collection of UnitTestFramework - compatible asserts defined for this type of node.
- int SerialNumber `[get]`

  Holds the device serial number.
- string NodeIdName `[get]`

  Get string representation of this node Id (NodeId).
- string NodeName `[get]`

  Get node name of this node.

A.26.1 Detailed Description

Represents single MATES-DIOX-MK1 node.

The DioxNode constructor is not intended to be called directly. Use Mates.New↩ DioxNode instead.

A.26.2 Member Function Documentation

void SetDout (
int *channel*,
bool *value*) `[inherited]`

Set value of an individual digital output.

Parameters

channel	The DOUT channel number [0,19].
value	The output value.

Example usage (mates_test_06.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_06
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node node = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               foreach (var val in new bool[]{false, true, false})
               {
```

```
                    // Assume loopback is pressent.
                    node.SetDout(0, val);
                    node.Assert.InputEquals(0, val);
                }
            }
            catch
            {
                Console.Write("Cannot discover node.\n");
            }
        }
    }
  }
}
```

void SetDoutAll (
int *value*) [inherited]

Set value of all digital outputs.

Parameters

value	The output value, LSB is OUT01.

Example usage (mates_test_05.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_05
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Assume there is loopback present between outputs and inputs.
               dio.SetDoutAll(0xCAFE);
               Assert.AreEqual(dio.GetDinAll(), 0xCAFE);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

bool GetDout (
int *channel*) [inherited]

Get value of an individual digital output.

Parameters

channel	The DOUT channel number [0,19].

Returns The result.

Example usage (mates_test_07.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
```

```
public class mates_test_07
{
   [TestMethod]
   public void TestMethod()
   {
      using (Mates mates = new Mates("mates_REMOTE.mon", 1))
      {
         try
         {
            Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
            // Set and verify.
            dio.SetDout(10, true);
            Assert.AreEqual(dio.GetDout(10), true);
            dio.SetDout(10, false);
            Assert.AreEqual(dio.GetDout(10), false);
         }
         catch
         {
            Console.Write("Cannot discover node.\n");
         }
      }
   }
}
```

int GetDoutAll ( ) [inherited]

Get value of all digital outputs.

Returns The result, LSB is OUT01.

Example usage (mates_test_08.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_08
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // No more than 20 outputs can be high.
               Assert.IsTrue(dio.GetDoutAll() <= 0xFFFFF);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

bool GetDin (int *channel*) [inherited]

Get value of an individual digital input.

Parameters

channel	The DIN channel number [0,19].

Returns The result.

Example usage (mates_test_09.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_09
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Enumerate all inputs.
               for (Mates.MatesInput input = Mates.MatesInput.IN01;
                    input <= Mates.MatesInput.IN20;
                    input++)
               {
                  Console.Write("Input {0}: {1}\n", input, dio.GetDin((int)input));
               }
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

int GetDinAll ( ) [inherited]

Get value of all digital inputs.

Returns The result, LSB is IN01.

Example usage (mates_test_10.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_10
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Send 8 of inputs to the outputs.
               int output = dio.GetDoutAll();
               int input = dio.GetDinAll();
               dio.SetDoutAll((output & 0xFFF00) | (input & 0xFF));
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

void InputSet (
int *channel*) [protected], [inherited]

Asserts that the specified input channel is in high state.

Parameters

channel	The input channel number [0, 19].

Implements IDioAsserts.

void OutputSet (
int *channel*) [protected], [inherited]

Asserts that the specified output channel is in high state.

Parameters

channel	The output channel number [0, 19].

Implements IDioAsserts.

void InputEquals (
int *channel,*
bool *state*) [protected], [inherited]

Asserts that the specified input channel is in the given state.

Parameters

channel	The input channel number [0, 19].
state	The expected state.

Implements IDioAsserts.

bool IsCanBridge ( ) [inherited]

Check if this node acts as a CAN bridge (it is connected directly to PC).

Returns True if CAN bridge, otherwise false.

int GetStatusRegister ( ) [inherited]

Get the value of the status register.

Returns The REG_COMMON_SR register value.

int GetRegisterI (
RegisterNumber *reg*) [inherited]

Get regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

float GetRegisterF (
RegisterNumber *reg*) [inherited]

Get floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

void SetRegisterI (
RegisterNumber *reg*,
int *val*) [inherited]

Set regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void SetRegisterF (
RegisterNumber *reg*,
float *val*) [inherited]

Set floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void RegisterEquals (
RegisterNumber *register*,
int *value*) [inherited]

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void RegisterEquals (
RegisterNumber *register*,
float *value*) [inherited]

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.

value	The expected value.

Implements INodeAsserts.

void LampTest ( ) [inherited]

Execute lamp test.

Example usage (mates_test_11.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_11
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Locate node by blinking its LEDs.
               dio.LampTest();
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

string NodeInfo ( ) [inherited]

Get node information as string.

Returns The node information.

Example usage (mates_test_13.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_13
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode dio = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
               // Get node info and print to console.
               string info = dio.NodeInfo();
               Console.Write(info);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
```

```
    }
}
```

A.26.3 Field Documentation

RegList Registers `[inherited]`

The registers that are associated with this node. Populated by the node constructor.

A.26.4 Property Documentation

IDioAsserts Assert `[get]`, `[inherited]`

Represents collection of UnitTestFramework - compatible asserts defined for this type of node.

int SerialNumber `[get]`, `[inherited]`

Holds the device serial number.

string NodeIdName `[get]`, `[inherited]`

Get string representation of this node Id (NodeId).

string NodeName `[get]`, `[inherited]`

Get node name of this node.

The documentation for this class was generated from the following files:

- DioxNode.cs
- MatesTypes.cs

A.27 Dac5Node Class Reference

Represents single MATES-DAC5-MK1 node.

Public Member Functions

- override double GetDacMaximum (int channel)

  Get the maximum allowable value of the DAC.
- override double GetDacMinimum (int channel)

  Get the minimum allowable value of the DAC.
- void SetDac (int channel, double voltage)

  Set the value of the DAC. Takes calibration data into account.
- void SetDacRaw (int channel, int val)

  Set raw value of the DAC.
- double GetDac (int channel)

  Get the value of the DAC.
- bool IsCanBridge ()

  Check if this node acts as a CAN bridge (it is connected directly to PC).
- int GetStatusRegister ()

Get the value of the status register.

- int GetRegisterI (RegisterNumber reg)

  Get regular (integral) MATES register.
- float GetRegisterF (RegisterNumber reg)

  Get floating point MATES register.
- void SetRegisterI (RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (RegisterNumber reg, float val)

  Set floating point MATES register.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.
- void LampTest ()

  Execute lamp test.
- string NodeInfo ()

  Get node information as string.

Data Fields

- RegList Registers

  The registers that are associated with this node. Populated by the node constructor.

Properties

- int SerialNumber `[get]`

  Holds the device serial number.
- string NodeIdName `[get]`

  Get string representation of this node Id (NodeId).
- string NodeName `[get]`

  Get node name of this node.

A.27.1 Detailed Description

Represents single MATES-DAC5-MK1 node.

The Dac5Node constructor is not intended to be called directly. Use Mates.New↩ Dac5Node instead.

A.27.2 Member Function Documentation

override double GetDacMaximum (
int *channel*) `[virtual]`

Get the maximum allowable value of the DAC.

Parameters

channel	The DAC channel number [0,39]

Returns The DAC maximum value in Volts.

Implements DacNode.

override double GetDacMinimum (
int *channel*) `[virtual]`

Get the minimum allowable value of the DAC.

Parameters

channel	The DAC channel number [0,39]

Returns The DAC minimum value in Volts.

Implements DacNode.

void SetDac (
int *channel,*
double *voltage*) `[inherited]`

Set the value of the DAC. Takes calibration data into account.

Parameters

channel	The DAC channel number [0,39]
voltage	The voltage value to set (Volts).

Example usage (mates_test_03.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_03
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dac5Node dac = mates.NewDac5Node(NodeId.mates_dac5_mk1_1);
               Adc5Node adc = mates.NewAdc5Node(NodeId.mates_adc5_mk1_1);
               // Assume there is loopback between DAC and ADC.
               dac.SetDac(5, 1.0);
               double check = adc.GetAdc(5);
               Assert.IsTrue(check >= 0.99 && check <= 1.01);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

Implements IDacNode.

void SetDacRaw (
int *channel,*
int *val*) [inherited]

Set raw value of the DAC.

Parameters

channel	The DAC channel number [0,39].
val	The raw DAC binary code to write.

Example usage (mates_test_04.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_04
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dac5Node dac = mates.NewDac5Node(NodeId.mates_dac5_mk1_1);
               // Set the first channel to the half of the range.
               dac.SetDacRaw(0, 65536 / 2);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

Implements IDacNode.

double GetDac (
int *channel*) [inherited]

Get the value of the DAC.

Parameters

channel	The DAC channel number [0,39]

Returns The DAC value in Volts.

Implements IDacNode.

bool IsCanBridge ( ) [inherited]

Check if this node acts as a CAN bridge (it is connected directly to PC).

Returns True if CAN bridge, otherwise false.

int GetStatusRegister ( ) [inherited]

Get the value of the status register.

Returns The REG_COMMON_SR register value.

int GetRegisterI (
RegisterNumber *reg*) [inherited]

Get regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

float GetRegisterF (
RegisterNumber *reg*) [inherited]

Get floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

void SetRegisterI (
RegisterNumber *reg*,
int *val*) [inherited]

Set regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void SetRegisterF (
RegisterNumber *reg*,
float *val*) [inherited]

Set floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void RegisterEquals (
RegisterNumber *register*,
int *value*) [inherited]

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void RegisterEquals (
RegisterNumber *register*,
float *value*) [inherited]

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void LampTest ( ) [inherited]

Execute lamp test.

Example usage (mates_test_11.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_11
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Locate node by blinking its LEDs.
               dio.LampTest();
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

string NodeInfo ( ) [inherited]

Get node information as string.

Returns The node information.

Example usage (mates_test_13.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_13
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode dio = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
```

```
            // Get node info and print to console.
            string info = dio.NodeInfo();
            Console.Write(info);
        }
        catch
        {
            Console.Write("Cannot discover node.\n");
        }
    }
  }
}
```

A.27.3 Field Documentation

RegList Registers `[inherited]`

The registers that are associated with this node. Populated by the node constructor.

A.27.4 Property Documentation

int SerialNumber `[get]`, `[inherited]`

Holds the device serial number.

string NodeIdName `[get]`, `[inherited]`

Get string representation of this node Id (NodeId).

string NodeName `[get]`, `[inherited]`

Get node name of this node.

The documentation for this class was generated from the following files:

- Dac5Node.cs
- MatesTypes.cs

A.28 Adc5Node Class Reference

Represents single MATES-ADC5-MK1 node.

Public Member Functions

- override int GetNumChannels ()

  Get number of channels of the ADC.
- override PhysicalValue GetRangeMax (int channel)

  Get the maximum value of a channel.
- double GetAdc (int channel)

  Get the value of the ADC.
- int GetAdcRaw (int channel)

  Get the raw value of the ADC.
- bool IsCanBridge ()

  Check if this node acts as a CAN bridge (it is connected directly to PC).
- int GetStatusRegister ()

  Get the value of the status register.

- int GetRegisterI (RegisterNumber reg)

  Get regular (integral) MATES register.
- float GetRegisterF (RegisterNumber reg)

  Get floating point MATES register.
- void SetRegisterI (RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (RegisterNumber reg, float val)

  Set floating point MATES register.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.
- void LampTest ()

  Execute lamp test.
- string NodeInfo ()

  Get node information as string.

Data Fields

- RegList Registers

  The registers that are associated with this node. Populated by the node constructor.

Properties

- int SerialNumber `[get]`

  Holds the device serial number.
- string NodeIdName `[get]`

  Get string representation of this node Id (NodeId).
- string NodeName `[get]`

  Get node name of this node.

A.28.1 Detailed Description

Represents single MATES-ADC5-MK1 node.

The Adc5Node constructor is not intended to be called directly. Use Mates.New↩DioxNode instead.

A.28.2 Member Function Documentation

override int GetNumChannels ( ) `[virtual]`

Get number of channels of the ADC.

Returns The number of channels.

Implements AdcNode.

override PhysicalValue GetRangeMax (
int *channel*) [virtual]

Get the maximum value of a channel.

Parameters

channel	

Returns The maximum value in Volts, Amperes or Degrees Centigrade.

Implements AdcNode.

double GetAdc (
int *channel*) [inherited]

Get the value of the ADC.

Parameters

channel	The ADC channel number [0,39].

Returns ADC voltage value in Volts.

Example usage (mates_test_01.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_01
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Adc5Node node = mates.NewAdc5Node(NodeId.mates_adc5_mk1_1);
               Console.Write("Voltage at channel IN01: {0} V\n", node.GetAdc(0));
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

Implements IAdcNode.

int GetAdcRaw (
int *channel*) [inherited]

Get the raw value of the ADC.

Parameters

channel	The ADC channel number [0,39]

Returns The raw ADC value.

Example usage (mates_test_02.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;
```

```
namespace unit_tests
{
   [TestClass]
   public class mates_test_02
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Adc5Node node = mates.NewAdc5Node(NodeId.mates_adc5_mk1_1);
               int val = node.GetAdcRaw(0);
               Console.Write("Raw value at channel IN01: {0} V\n", val);
               Assert.IsTrue(val >= 0 && val <= 65535);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

Implements IAdcNode.

bool IsCanBridge ( ) [inherited]

Check if this node acts as a CAN bridge (it is connected directly to PC).

Returns True if CAN bridge, otherwise false.

int GetStatusRegister ( ) [inherited]

Get the value of the status register.

Returns The REG_COMMON_SR register value.

int GetRegisterI (RegisterNumber *reg*) [inherited]

Get regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

float GetRegisterF (RegisterNumber *reg*) [inherited]

Get floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

void SetRegisterI (
RegisterNumber *reg*,
int *val*) [inherited]

Set regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void SetRegisterF (
RegisterNumber *reg*,
float *val*) [inherited]

Set floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void RegisterEquals (
RegisterNumber *register*,
int *value*) [inherited]

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void RegisterEquals (
RegisterNumber *register*,
float *value*) [inherited]

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void LampTest ( ) [inherited]

Execute lamp test.

Example usage (mates_test_11.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
```

```
public class mates_test_11
{
   [TestMethod]
   public void TestMethod()
   {
      using (Mates mates = new Mates("mates_REMOTE.mon", 1))
      {
         try
         {
            Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
            // Locate node by blinking its LEDs.
            dio.LampTest();
         }
         catch
         {
            Console.Write("Cannot discover node.\n");
         }
      }
   }
}
```

string NodeInfo ( ) [inherited]

Get node information as string.

Returns The node information.

Example usage (mates_test_13.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_13
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode dio = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
               // Get node info and print to console.
               string info = dio.NodeInfo();
               Console.Write(info);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

A.28.3 Field Documentation

RegList Registers [inherited]

The registers that are associated with this node. Populated by the node constructor.

A.28.4 Property Documentation

int SerialNumber [get], [inherited]

Holds the device serial number.

string NodeIdName `[get], [inherited]`

Get string representation of this node Id (NodeId).

string NodeName `[get], [inherited]`

Get node name of this node.

The documentation for this class was generated from the following files:

- Adc5Node.cs
- MatesTypes.cs

A.29 UccNode Class Reference

Represents single MATES-UCC-MK1 node.

Public Member Functions

- void AlcdWrite (int row, int col, string text)

  Write text onto the alphanumeric LCD display.
- double GetAdc (int channel)

  Get the value of the ADC.
- int GetAdcRaw (int channel)

  Get the raw value of the ADC.
- int GetNumChannels ()

  Get number of channels of the ADC.
- PhysicalValue GetRangeMax (int channel)

  Get the maximum value of a channel.
- void SetDout (int channel, bool value)

  Set value of an individual digital output.
- void SetDoutAll (int value)

  Set value of all digital outputs.
- bool GetDout (int channel)

  Get value of an individual digital output.
- int GetDoutAll ()

  Get value of all digital outputs.
- bool GetDin (int channel)

  Get value of an individual digital input.
- int GetDinAll ()

  Get value of all digital inputs.
- bool IsCanBridge ()

  Check if this node acts as a CAN bridge (it is connected directly to PC).
- int GetStatusRegister ()

  Get the value of the status register.
- int GetRegisterI (RegisterNumber reg)

  Get regular (integral) MATES register.

- float GetRegisterF (RegisterNumber reg)

  Get floating point MATES register.
- void SetRegisterI (RegisterNumber reg, int val)

  Set regular (integral) MATES register.
- void SetRegisterF (RegisterNumber reg, float val)

  Set floating point MATES register.
- void RegisterEquals (RegisterNumber register, int value)

  Asserts that the specified integral register has the specified value.
- void RegisterEquals (RegisterNumber register, float value)

  Asserts that the specified floating point register has the specified value.
- void LampTest ()

  Execute lamp test.
- string NodeInfo ()

  Get node information as string.

Data Fields

- RegList Registers

  The registers that are associated with this node. Populated by the node constructor.

Protected Member Functions

- void InputSet (int channel)

  Asserts that the specified input channel is in high state.
- void OutputSet (int channel)

  Asserts that the specified output channel is in high state.
- void InputEquals (int channel, bool state)

  Asserts that the specified input channel is in the given state.

Properties

- IDioAsserts Assert `[get]`

  Represents collection of UnitTestFramework - compatible asserts defined for this type of node.
- int SerialNumber `[get]`

  Holds the device serial number.
- string NodeIdName `[get]`

  Get string representation of this node Id (NodeId).
- string NodeName `[get]`

  Get node name of this node.

A.29.1 Detailed Description

Represents single MATES-UCC-MK1 node.

The UccNode constructor is not intended to be called directly. Use Mates.New↩ UccNode instead.

A.29.2 Member Function Documentation

void AlcdWrite (
int *row*,
int *col*,
string *text*)

Write text onto the alphanumeric LCD display.

Parameters

row	The LCD row number (0 - based).
col	The LCD column number (0 - based).
text	The text to write at the given position.

Example usage (mates_test_14.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_14
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode ucc = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
               ucc.AlcdWrite(0, 0, "CS test!");
            }
            catch (Exception ex)
            {
               Console.Write("Cannot discover node: " + ex + "\n");
            }
         }
      }
   }
}
```

double GetAdc (
int *channel*)

Get the value of the ADC.

Parameters

channel	The ADC channel number [0,39].

Returns ADC voltage value in Volts.

Example usage (mates_test_01.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_01
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Adc5Node node = mates.NewAdc5Node(NodeId.mates_adc5_mk1_1);
```

```
                Console.Write("Voltage at channel IN01: {0} V\n", node.GetAdc(0));
            }
            catch
            {
                Console.Write("Cannot discover node.\n");
            }
        }
    }
  }
}
```

Implements IAdcNode.

int GetAdcRaw (
int *channel*)

Get the raw value of the ADC.

Parameters

channel	The ADC channel number [0,39]

Returns The raw ADC value.

Example usage (mates_test_02.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_02
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Adc5Node node = mates.NewAdc5Node(NodeId.mates_adc5_mk1_1);
               int val = node.GetAdcRaw(0);
               Console.Write("Raw value at channel IN01: {0} V\n", val);
               Assert.IsTrue(val >= 0 && val <= 65535);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

Implements IAdcNode.

int GetNumChannels ( )

Get number of channels of the ADC.

Returns The number of channels.

Implements IAdcNode.

PhysicalValue GetRangeMax (
int *channel*)

Get the maximum value of a channel.

Parameters

channel	

Returns The maximum value in Volts, Amperes or Degrees Centigrade.

Implements IAdcNode.

void SetDout (
int *channel*,
bool *value*) [inherited]

Set value of an individual digital output.

Parameters

channel	The DOUT channel number [0,19].
value	The output value.

Example usage (mates_test_06.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_06
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node node = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               foreach (var val in new bool[]{false, true, false})
               {
                  // Assume loopback is pressent.
                  node.SetDout(0, val);
                  node.Assert.InputEquals(0, val);
               }
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

void SetDoutAll (
int *value*) [inherited]

Set value of all digital outputs.

Parameters

value	The output value, LSB is OUT01.

Example usage (mates_test_05.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_05
   {
      [TestMethod]
```

```
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Assume there is loopback present between outputs and inputs.
               dio.SetDoutAll(0xCAFE);
               Assert.AreEqual(dio.GetDinAll(), 0xCAFE);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

bool GetDout (
int *channel*) [inherited]

Get value of an individual digital output.

Parameters

channel	The DOUT channel number [0,19].

Returns The result.

Example usage (mates_test_07.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_07
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Set and verify.
               dio.SetDout(10, true);
               Assert.AreEqual(dio.GetDout(10), true);
               dio.SetDout(10, false);
               Assert.AreEqual(dio.GetDout(10), false);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

int GetDoutAll ( ) [inherited]

Get value of all digital outputs.

Returns The result, LSB is OUT01.

Example usage (mates_test_08.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
```

```
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_08
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // No more than 20 outputs can be high.
               Assert.IsTrue(dio.GetDoutAll() <= 0xFFFFF);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

bool GetDin (
int *channel*) [inherited]

Get value of an individual digital input.

Parameters

channel	The DIN channel number [0,19].

Returns The result.

Example usage (mates_test_09.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_09
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Enumerate all inputs.
               for (Mates.MatesInput input = Mates.MatesInput.IN01;
                    input <= Mates.MatesInput.IN20;
                    input++)
               {
                  Console.Write("Input {0}: {1}\n", input, dio.GetDin((int)input));
               }
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

int GetDinAll ( ) [inherited]

Get value of all digital inputs.

Returns The result, LSB is IN01.

Example usage (mates_test_10.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_10
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Send 8 of inputs to the outputs.
               int output = dio.GetDoutAll();
               int input = dio.GetDinAll();
               dio.SetDoutAll((output & 0xFFF00) | (input & 0xFF));
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

void InputSet (
int *channel*) [protected], [inherited]

Asserts that the specified input channel is in high state.

Parameters

channel	The input channel number [0, 19].

Implements IDioAsserts.

void OutputSet (
int *channel*) [protected], [inherited]

Asserts that the specified output channel is in high state.

Parameters

channel	The output channel number [0, 19].

Implements IDioAsserts.

void InputEquals (
int *channel,*
bool *state*) [protected], [inherited]

Asserts that the specified input channel is in the given state.

Parameters

channel	The input channel number [0, 19].
state	The expected state.

Implements IDioAsserts.

bool IsCanBridge ( ) [inherited]

Check if this node acts as a CAN bridge (it is connected directly to PC).

Returns True if CAN bridge, otherwise false.

int GetStatusRegister ( ) [inherited]

Get the value of the status register.

Returns The REG_COMMON_SR register value.

int GetRegisterI (
RegisterNumber *reg*) [inherited]

Get regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

float GetRegisterF (
RegisterNumber *reg*) [inherited]

Get floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).

Returns The register value.

void SetRegisterI (
RegisterNumber *reg*,
int *val*) [inherited]

Set regular (integral) MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void SetRegisterF (
RegisterNumber *reg*,
float *val*) `[inherited]`

Set floating point MATES register.

Parameters

reg	The register number to access (see Viresco.Mates.Registers.↩ RegisterNumber).
val	The register value.

void RegisterEquals (
RegisterNumber *register*,
int *value*) [inherited]

Asserts that the specified integral register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void RegisterEquals (
RegisterNumber *register*,
float *value*) [inherited]

Asserts that the specified floating point register has the specified value.

Parameters

register	The register number.
value	The expected value.

Implements INodeAsserts.

void LampTest ( ) [inherited]

Execute lamp test.

Example usage (mates_test_11.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_11
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               Dio3Node dio = mates.NewDio3Node(NodeId.mates_dio3_mk1_1);
               // Locate node by blinking its LEDs.
               dio.LampTest();
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

string NodeInfo ( ) [inherited]

Get node information as string.

Returns The node information.

Example usage (mates_test_13.cs):

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Viresco.Mates;

namespace unit_tests
{
   [TestClass]
   public class mates_test_13
   {
      [TestMethod]
      public void TestMethod()
      {
         using (Mates mates = new Mates("mates_REMOTE.mon", 1))
         {
            try
            {
               UccNode dio = mates.NewUccNode(NodeId.mates_ucc_mk1_1);
               // Get node info and print to console.
               string info = dio.NodeInfo();
               Console.Write(info);
            }
            catch
            {
               Console.Write("Cannot discover node.\n");
            }
         }
      }
   }
}
```

A.29.3 Field Documentation

RegList Registers [inherited]

The registers that are associated with this node. Populated by the node constructor.

A.29.4 Property Documentation

IDioAsserts Assert [get], [inherited]

Represents collection of UnitTestFramework - compatible asserts defined for this type of node.

int SerialNumber [get], [inherited]

Holds the device serial number.

string NodeIdName [get], [inherited]

Get string representation of this node Id (NodeId).

string NodeName [get], [inherited]

Get node name of this node.

The documentation for this class was generated from the following files:

- UccNode.cs

- MatesTypes.cs

A.30 mates_regs.h File Reference

MATES nodes registers definitions.

Enumerations

A.30.1 Detailed Description

MATES nodes registers definitions.

See MATES User's manual for registers documentation.

A.30.2 Enumeration Type Documentation

enum MATES_REG_TYPE

MATES register enumeration. Used to access MATES registers (both integral and floating point).

Enumerator *REG_COMMON_TEST* : (0) TEST (#0x0, RWN, init.: 0xFF, min.: 0x0, max.: 0xFF↩ FFFFFF) - Used for internal testing, do not use.

REG_COMMON_DIR : (1) DIR (#0x1, RO, init.: 0x0) - Device identification register

REG_COMMON_BIR : (2) BIR (#0x2, RO, init.: 0x0) - Boot loader identification register

REG_COMMON_FIR : (3) FIR (#0x3, RO, init.: 0x0) - Firmware identification register

REG_COMMON_SR : (4) SR (#0x4, RO, init.: 0x0) - Status register

REG_COMMON_CR : (5) CR (#0x5, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Control register

REG_COMMON_TUT : (6) TUT (#0x6, RO, init.: 0x0) - Total up time register

REG_COMMON_CUT : (7) CUT (#0x7, RO, init.: 0x0) - Current up time register

REG_COMMON_FUT : (8) FUT (#0x8, RO, init.: 0x0) - Firmware up time register

REG_COMMON_DPD : (9) DPD (#0x9, RO, init.: 0x0) - Device production date register

REG_COMMON_PUC : (10) PUC (#0xA, RO, init.: 0x0) - Power up counter register

REG_COMMON_HIR : (11) HIR (#0xB, RO, init.: 0x0) - Hardware identification register

REG_COMMON_CAPS01 : (12) CAPS01 (#0xC, RO, init.: 0x0) - Device capabilities register #1

REG_COMMON_CAPS02 : (13) CAPS02 (#0xD, RO, init.: 0x0) - Device capabilities register #2

REG_MATES_DIO3_MK1_ODEF : (512) ODEF (#0x200, RWN, init.: 0x0, min.: 0x0, max.: 0x800FFFFF) - Outputs default value register

REG_MATES_DIO3_MK1_ODIS : (513) ODIS (#0x201, RW, init.: 0x0, min.: 0x0, max.: 0xF↩ FFFF) - Outputs disable register

REG_MATES_DIO3_MK1_PRD01 : (514) PRD01 (#0x202, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 01 period register

REG_MATES_DIO3_MK1_PRD02 : (515) PRD02 (#0x203, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 02 period register

REG_MATES_DIO3_MK1_PRD03 : (516) PRD03 (#0x204, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 03 period register

REG_MATES_DIO3_MK1_PRD04 : (517) PRD04 (#0x205, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 04 period register

REG_MATES_DIO3_MK1_PRD05 : (518) PRD05 (#0x206, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 05 period register

REG_MATES_DIO3_MK1_PRD06 : (519) PRD06 (#0x207, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 06 period register

REG_MATES_DIO3_MK1_PRD07 : (520) PRD07 (#0x208, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 07 period register

REG_MATES_DIO3_MK1_PRD08 : (521) PRD08 (#0x209, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 08 period register

REG_MATES_DIO3_MK1_PRD09 : (522) PRD09 (#0x20A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 09 period register

REG_MATES_DIO3_MK1_PRD10 : (523) PRD10 (#0x20B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 10 period register

REG_MATES_DIO3_MK1_PRD11 : (524) PRD11 (#0x20C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 11 period register

REG_MATES_DIO3_MK1_PRD12 : (525) PRD12 (#0x20D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 12 period register

REG_MATES_DIO3_MK1_PRD13 : (526) PRD13 (#0x20E, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 13 period register

REG_MATES_DIO3_MK1_PRD14 : (527) PRD14 (#0x20F, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 14 period register

REG_MATES_DIO3_MK1_PRD15 : (528) PRD15 (#0x210, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 15 period register

REG_MATES_DIO3_MK1_PRD16 : (529) PRD16 (#0x211, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 16 period register

REG_MATES_DIO3_MK1_PRD17 : (530) PRD17 (#0x212, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 17 period register

REG_MATES_DIO3_MK1_PRD18 : (531) PRD18 (#0x213, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 18 period register

REG_MATES_DIO3_MK1_PRD19 : (532) PRD19 (#0x214, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 19 period register

REG_MATES_DIO3_MK1_PRD20 : (533) PRD20 (#0x215, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 20 period register

REG_MATES_DIO3_MK1_TRISE01 : (534) TRISE01 (#0x216, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 01 rise time register

REG_MATES_DIO3_MK1_TRISE02 : (535) TRISE02 (#0x217, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 02 rise time register

REG_MATES_DIO3_MK1_TRISE03 : (536) TRISE03 (#0x218, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 03 rise time register

REG_MATES_DIO3_MK1_TRISE04 : (537) TRISE04 (#0x219, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 04 rise time register

REG_MATES_DIO3_MK1_TRISE05 : (538) TRISE05 (#0x21A, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 05 rise time register

REG_MATES_DIO3_MK1_TRISE06 : (539) TRISE06 (#0x21B, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 06 rise time register

REG_MATES_DIO3_MK1_TRISE07 : (540) TRISE07 (#0x21C, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 07 rise time register

REG_MATES_DIO3_MK1_TRISE08 : (541) TRISE08 (#0x21D, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 08 rise time register

REG_MATES_DIO3_MK1_TRISE09 : (542) TRISE09 (#0x21E, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 09 rise time register

REG_MATES_DIO3_MK1_TRISE10 : (543) TRISE10 (#0x21F, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 10 rise time register

REG_MATES_DIO3_MK1_TRISE11 : (544) TRISE11 (#0x220, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 11 rise time register

REG_MATES_DIO3_MK1_TRISE12 : (545) TRISE12 (#0x221, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 12 rise time register

REG_MATES_DIO3_MK1_TRISE13 : (546) TRISE13 (#0x222, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 13 rise time register

REG_MATES_DIO3_MK1_TRISE14 : (547) TRISE14 (#0x223, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 14 rise time register

REG_MATES_DIO3_MK1_TRISE15 : (548) TRISE15 (#0x224, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 15 rise time register

REG_MATES_DIO3_MK1_TRISE16 : (549) TRISE16 (#0x225, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 16 rise time register

REG_MATES_DIO3_MK1_TRISE17 : (550) TRISE17 (#0x226, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 17 rise time register

REG_MATES_DIO3_MK1_TRISE18 : (551) TRISE18 (#0x227, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 18 rise time register

REG_MATES_DIO3_MK1_TRISE19 : (552) TRISE19 (#0x228, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 19 rise time register

REG_MATES_DIO3_MK1_TRISE20 : (553) TRISE20 (#0x229, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 20 rise time register

REG_MATES_DIO3_MK1_TFALL01 : (554) TFALL01 (#0x22A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 01 fall time register

REG_MATES_DIO3_MK1_TFALL02 : (555) TFALL02 (#0x22B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 02 fall time register

REG_MATES_DIO3_MK1_TFALL03 : (556) TFALL03 (#0x22C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 03 fall time register

REG_MATES_DIO3_MK1_TFALL04 : (557) TFALL04 (#0x22D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 04 fall time register

REG_MATES_DIO3_MK1_TFALL05 : (558) TFALL05 (#0x22E, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 05 fall time register

REG_MATES_DIO3_MK1_TFALL06 : (559) TFALL06 (#0x22F, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 06 fall time register

REG_MATES_DIO3_MK1_TFALL07 : (560) TFALL07 (#0x230, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 07 fall time register

REG_MATES_DIO3_MK1_TFALL08 : (561) TFALL08 (#0x231, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 08 fall time register

REG_MATES_DIO3_MK1_TFALL09 : (562) TFALL09 (#0x232, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 09 fall time register

REG_MATES_DIO3_MK1_TFALL10 : (563) TFALL10 (#0x233, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 10 fall time register

REG_MATES_DIO3_MK1_TFALL11 : (564) TFALL11 (#0x234, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 11 fall time register

REG_MATES_DIO3_MK1_TFALL12 : (565) TFALL12 (#0x235, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 12 fall time register

REG_MATES_DIO3_MK1_TFALL13 : (566) TFALL13 (#0x236, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 13 fall time register

REG_MATES_DIO3_MK1_TFALL14 : (567) TFALL14 (#0x237, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 14 fall time register

REG_MATES_DIO3_MK1_TFALL15 : (568) TFALL15 (#0x238, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 15 fall time register

REG_MATES_DIO3_MK1_TFALL16 : (569) TFALL16 (#0x239, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 16 fall time register

REG_MATES_DIO3_MK1_TFALL17 : (570) TFALL17 (#0x23A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 17 fall time register

REG_MATES_DIO3_MK1_TFALL18 : (571) TFALL18 (#0x23B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 18 fall time register

REG_MATES_DIO3_MK1_TFALL19 : (572) TFALL19 (#0x23C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 19 fall time register

REG_MATES_DIO3_MK1_TFALL20 : (573) TFALL20 (#0x23D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 20 fall time register

REG_MATES_DIO3_MK1_ORP : (574) ORP (#0x23E, RWN, init.: 0x0, min.: 0x0, max.: 0x↩ FFFFFFFF) - Outputs restore period

REG_MATES_DIO3_MK1_ORRT : (575) ORRT (#0x23F, RO, init.: 0x0) - Outptus restore remaining time register

REG_MATES_DIO3_MK1_PSEL : (576) PSEL (#0x240, RWN, init.: 0x0, min.: 0x0, max.: 0x↩ FFFFF) - Pull-up / pull-down selection register

REG_MATES_DIO3_MK1_CANID : (577) CANID (#0x241, RWN, init.: 20, min.: 20, max.: 29) - The node's CAN ID / address register

REG_MATES_DAC5_MK1_DEF01 : (1280) DEF01 (#0x500, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT01 default voltage register

REG_MATES_DAC5_MK1_DEF02 : (1281) DEF02 (#0x501, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT02 default voltage register

REG_MATES_DAC5_MK1_DEF03 : (1282) DEF03 (#0x502, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT03 default voltage register

REG_MATES_DAC5_MK1_DEF04 : (1283) DEF04 (#0x503, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT04 default voltage register

REG_MATES_DAC5_MK1_DEF05 : (1284) DEF05 (#0x504, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT05 default voltage register

REG_MATES_DAC5_MK1_DEF06 : (1285) DEF06 (#0x505, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT06 default voltage register

REG_MATES_DAC5_MK1_DEF07 : (1286) DEF07 (#0x506, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT07 default voltage register

REG_MATES_DAC5_MK1_DEF08 : (1287) DEF08 (#0x507, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT08 default voltage register

REG_MATES_DAC5_MK1_DEF09 : (1288) DEF09 (#0x508, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT09 default voltage register

REG_MATES_DAC5_MK1_DEF10 : (1289) DEF10 (#0x509, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT10 default voltage register

REG_MATES_DAC5_MK1_DEF11 : (1290) DEF11 (#0x50A, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT11 default voltage register

REG_MATES_DAC5_MK1_DEF12 : (1291) DEF12 (#0x50B, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT12 default voltage register

REG_MATES_DAC5_MK1_DEF13 : (1292) DEF13 (#0x50C, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT13 default voltage register

REG_MATES_DAC5_MK1_DEF14 : (1293) DEF14 (#0x50D, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT14 default voltage register

REG_MATES_DAC5_MK1_DEF15 : (1294) DEF15 (#0x50E, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT15 default voltage register

REG_MATES_DAC5_MK1_DEF16 : (1295) DEF16 (#0x50F, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT16 default voltage register

REG_MATES_DAC5_MK1_DEF17 : (1296) DEF17 (#0x510, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT17 default voltage register

REG_MATES_DAC5_MK1_DEF18 : (1297) DEF18 (#0x511, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT18 default voltage register

REG_MATES_DAC5_MK1_DEF19 : (1298) DEF19 (#0x512, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT19 default voltage register

REG_MATES_DAC5_MK1_DEF20 : (1299) DEF20 (#0x513, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT20 default voltage register

REG_MATES_DAC5_MK1_DEF21 : (1300) DEF21 (#0x514, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT21 default voltage register

REG_MATES_DAC5_MK1_DEF22 : (1301) DEF22 (#0x515, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT22 default voltage register

REG_MATES_DAC5_MK1_DEF23 : (1302) DEF23 (#0x516, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT23 default voltage register

REG_MATES_DAC5_MK1_DEF24 : (1303) DEF24 (#0x517, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT24 default voltage register

REG_MATES_DAC5_MK1_DEF25 : (1304) DEF25 (#0x518, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT25 default voltage register

REG_MATES_DAC5_MK1_DEF26 : (1305) DEF26 (#0x519, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT26 default voltage register

REG_MATES_DAC5_MK1_DEF27 : (1306) DEF27 (#0x51A, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT27 default voltage register

REG_MATES_DAC5_MK1_DEF28 : (1307) DEF28 (#0x51B, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT28 default voltage register

REG_MATES_DAC5_MK1_DEF29 : (1308) DEF29 (#0x51C, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT29 default voltage register

REG_MATES_DAC5_MK1_DEF30 : (1309) DEF30 (#0x51D, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT30 default voltage register

REG_MATES_DAC5_MK1_DEF31 : (1310) DEF31 (#0x51E, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT31 default voltage register

REG_MATES_DAC5_MK1_DEF32 : (1311) DEF32 (#0x51F, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT32 default voltage register

REG_MATES_DAC5_MK1_DEF33 : (1312) DEF33 (#0x520, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT33 default voltage register

REG_MATES_DAC5_MK1_DEF34 : (1313) DEF34 (#0x521, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT34 default voltage register

REG_MATES_DAC5_MK1_DEF35 : (1314) DEF35 (#0x522, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT35 default voltage register

REG_MATES_DAC5_MK1_DEF36 : (1315) DEF36 (#0x523, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT36 default voltage register

REG_MATES_DAC5_MK1_DEF37 : (1316) DEF37 (#0x524, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT37 default voltage register

REG_MATES_DAC5_MK1_DEF38 : (1317) DEF38 (#0x525, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT38 default voltage register

REG_MATES_DAC5_MK1_DEF39 : (1318) DEF39 (#0x526, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT39 default voltage register

REG_MATES_DAC5_MK1_DEF40 : (1319) DEF40 (#0x527, RWN, init.: 0.1, min.: 0.0, max.: 5.0) - OUT40 default voltage register

REG_MATES_DAC5_MK1_MIN01 : (1320) MIN01 (#0x528, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT01 minimum voltage register

REG_MATES_DAC5_MK1_MIN02 : (1321) MIN02 (#0x529, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT02 minimum voltage register

REG_MATES_DAC5_MK1_MIN03 : (1322) MIN03 (#0x52A, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT03 minimum voltage register

REG_MATES_DAC5_MK1_MIN04 : (1323) MIN04 (#0x52B, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT04 minimum voltage register

REG_MATES_DAC5_MK1_MIN05 : (1324) MIN05 (#0x52C, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT05 minimum voltage register

REG_MATES_DAC5_MK1_MIN06 : (1325) MIN06 (#0x52D, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT06 minimum voltage register

REG_MATES_DAC5_MK1_MIN07 : (1326) MIN07 (#0x52E, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT07 minimum voltage register

REG_MATES_DAC5_MK1_MIN08 : (1327) MIN08 (#0x52F, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT08 minimum voltage register

REG_MATES_DAC5_MK1_MIN09 : (1328) MIN09 (#0x530, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT09 minimum voltage register

REG_MATES_DAC5_MK1_MIN10 : (1329) MIN10 (#0x531, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT10 minimum voltage register

REG_MATES_DAC5_MK1_MIN11 : (1330) MIN11 (#0x532, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT11 minimum voltage register

REG_MATES_DAC5_MK1_MIN12 : (1331) MIN12 (#0x533, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT12 minimum voltage register

REG_MATES_DAC5_MK1_MIN13 : (1332) MIN13 (#0x534, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT13 minimum voltage register

REG_MATES_DAC5_MK1_MIN14 : (1333) MIN14 (#0x535, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT14 minimum voltage register

REG_MATES_DAC5_MK1_MIN15 : (1334) MIN15 (#0x536, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT15 minimum voltage register

REG_MATES_DAC5_MK1_MIN16 : (1335) MIN16 (#0x537, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT16 minimum voltage register

REG_MATES_DAC5_MK1_MIN17 : (1336) MIN17 (#0x538, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT17 minimum voltage register

REG_MATES_DAC5_MK1_MIN18 : (1337) MIN18 (#0x539, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT18 minimum voltage register

REG_MATES_DAC5_MK1_MIN19 : (1338) MIN19 (#0x53A, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT19 minimum voltage register

REG_MATES_DAC5_MK1_MIN20 : (1339) MIN20 (#0x53B, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT20 minimum voltage register

REG_MATES_DAC5_MK1_MIN21 : (1340) MIN21 (#0x53C, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT21 minimum voltage register

REG_MATES_DAC5_MK1_MIN22 : (1341) MIN22 (#0x53D, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT22 minimum voltage register

REG_MATES_DAC5_MK1_MIN23 : (1342) MIN23 (#0x53E, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT23 minimum voltage register

REG_MATES_DAC5_MK1_MIN24 : (1343) MIN24 (#0x53F, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT24 minimum voltage register

REG_MATES_DAC5_MK1_MIN25 : (1344) MIN25 (#0x540, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT25 minimum voltage register

REG_MATES_DAC5_MK1_MIN26 : (1345) MIN26 (#0x541, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT26 minimum voltage register

REG_MATES_DAC5_MK1_MIN27 : (1346) MIN27 (#0x542, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT27 minimum voltage register

REG_MATES_DAC5_MK1_MIN28 : (1347) MIN28 (#0x543, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT28 minimum voltage register

REG_MATES_DAC5_MK1_MIN29 : (1348) MIN29 (#0x544, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT29 minimum voltage register

REG_MATES_DAC5_MK1_MIN30 : (1349) MIN30 (#0x545, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT30 minimum voltage register

REG_MATES_DAC5_MK1_MIN31 : (1350) MIN31 (#0x546, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT31 minimum voltage register

REG_MATES_DAC5_MK1_MIN32 : (1351) MIN32 (#0x547, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT32 minimum voltage register

REG_MATES_DAC5_MK1_MIN33 : (1352) MIN33 (#0x548, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT33 minimum voltage register

REG_MATES_DAC5_MK1_MIN34 : (1353) MIN34 (#0x549, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT34 minimum voltage register

REG_MATES_DAC5_MK1_MIN35 : (1354) MIN35 (#0x54A, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT35 minimum voltage register

REG_MATES_DAC5_MK1_MIN36 : (1355) MIN36 (#0x54B, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT36 minimum voltage register

REG_MATES_DAC5_MK1_MIN37 : (1356) MIN37 (#0x54C, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT37 minimum voltage register

REG_MATES_DAC5_MK1_MIN38 : (1357) MIN38 (#0x54D, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT38 minimum voltage register

REG_MATES_DAC5_MK1_MIN39 : (1358) MIN39 (#0x54E, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT39 minimum voltage register

REG_MATES_DAC5_MK1_MIN40 : (1359) MIN40 (#0x54F, RWN, init.: 0.0, min.: 0.0, max.: 5.0) - OUT40 minimum voltage register

REG_MATES_DAC5_MK1_MAX01 : (1360) MAX01 (#0x550, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT01 maximum voltage register

REG_MATES_DAC5_MK1_MAX02 : (1361) MAX02 (#0x551, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT02 maximum voltage register

REG_MATES_DAC5_MK1_MAX03 : (1362) MAX03 (#0x552, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT03 maximum voltage register

REG_MATES_DAC5_MK1_MAX04 : (1363) MAX04 (#0x553, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT04 maximum voltage register

REG_MATES_DAC5_MK1_MAX05 : (1364) MAX05 (#0x554, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT05 maximum voltage register

REG_MATES_DAC5_MK1_MAX06 : (1365) MAX06 (#0x555, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT06 maximum voltage register

REG_MATES_DAC5_MK1_MAX07 : (1366) MAX07 (#0x556, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT07 maximum voltage register

REG_MATES_DAC5_MK1_MAX08 : (1367) MAX08 (#0x557, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT08 maximum voltage register

REG_MATES_DAC5_MK1_MAX09 : (1368) MAX09 (#0x558, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT09 maximum voltage register

REG_MATES_DAC5_MK1_MAX10 : (1369) MAX10 (#0x559, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT10 maximum voltage register

REG_MATES_DAC5_MK1_MAX11 : (1370) MAX11 (#0x55A, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT11 maximum voltage register

REG_MATES_DAC5_MK1_MAX12 : (1371) MAX12 (#0x55B, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT12 maximum voltage register

REG_MATES_DAC5_MK1_MAX13 : (1372) MAX13 (#0x55C, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT13 maximum voltage register

REG_MATES_DAC5_MK1_MAX14 : (1373) MAX14 (#0x55D, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT14 maximum voltage register

REG_MATES_DAC5_MK1_MAX15 : (1374) MAX15 (#0x55E, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT15 maximum voltage register

REG_MATES_DAC5_MK1_MAX16 : (1375) MAX16 (#0x55F, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT16 maximum voltage register

REG_MATES_DAC5_MK1_MAX17 : (1376) MAX17 (#0x560, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT17 maximum voltage register

REG_MATES_DAC5_MK1_MAX18 : (1377) MAX18 (#0x561, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT18 maximum voltage register

REG_MATES_DAC5_MK1_MAX19 : (1378) MAX19 (#0x562, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT19 maximum voltage register

REG_MATES_DAC5_MK1_MAX20 : (1379) MAX20 (#0x563, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT20 maximum voltage register

REG_MATES_DAC5_MK1_MAX21 : (1380) MAX21 (#0x564, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT21 maximum voltage register

REG_MATES_DAC5_MK1_MAX22 : (1381) MAX22 (#0x565, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT22 maximum voltage register

REG_MATES_DAC5_MK1_MAX23 : (1382) MAX23 (#0x566, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT23 maximum voltage register

REG_MATES_DAC5_MK1_MAX24 : (1383) MAX24 (#0x567, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT24 maximum voltage register

REG_MATES_DAC5_MK1_MAX25 : (1384) MAX25 (#0x568, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT25 maximum voltage register

REG_MATES_DAC5_MK1_MAX26 : (1385) MAX26 (#0x569, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT26 maximum voltage register

REG_MATES_DAC5_MK1_MAX27 : (1386) MAX27 (#0x56A, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT27 maximum voltage register

REG_MATES_DAC5_MK1_MAX28 : (1387) MAX28 (#0x56B, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT28 maximum voltage register

REG_MATES_DAC5_MK1_MAX29 : (1388) MAX29 (#0x56C, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT29 maximum voltage register

REG_MATES_DAC5_MK1_MAX30 : (1389) MAX30 (#0x56D, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT30 maximum voltage register

REG_MATES_DAC5_MK1_MAX31 : (1390) MAX31 (#0x56E, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT31 maximum voltage register

REG_MATES_DAC5_MK1_MAX32 : (1391) MAX32 (#0x56F, RWN, init.: 3.0, min.: 0.0, max.↩ : 5.0) - OUT32 maximum voltage register

REG_MATES_DAC5_MK1_MAX33 : (1392) MAX33 (#0x570, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT33 maximum voltage register

REG_MATES_DAC5_MK1_MAX34 : (1393) MAX34 (#0x571, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT34 maximum voltage register

REG_MATES_DAC5_MK1_MAX35 : (1394) MAX35 (#0x572, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT35 maximum voltage register

REG_MATES_DAC5_MK1_MAX36 : (1395) MAX36 (#0x573, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT36 maximum voltage register

REG_MATES_DAC5_MK1_MAX37 : (1396) MAX37 (#0x574, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT37 maximum voltage register

REG_MATES_DAC5_MK1_MAX38 : (1397) MAX38 (#0x575, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT38 maximum voltage register

REG_MATES_DAC5_MK1_MAX39 : (1398) MAX39 (#0x576, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT39 maximum voltage register

REG_MATES_DAC5_MK1_MAX40 : (1399) MAX40 (#0x577, RWN, init.: 3.0, min.: 0.0, max.: 5.0) - OUT40 maximum voltage register

REG_MATES_DAC5_MK1_COF01 : (1400) COF01 (#0x578, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT01 calibration offset register

REG_MATES_DAC5_MK1_COF02 : (1401) COF02 (#0x579, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT02 calibration offset register

REG_MATES_DAC5_MK1_COF03 : (1402) COF03 (#0x57A, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - OUT03 calibration offset register

REG_MATES_DAC5_MK1_COF04 : (1403) COF04 (#0x57B, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - OUT04 calibration offset register

REG_MATES_DAC5_MK1_COF05 : (1404) COF05 (#0x57C, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - OUT05 calibration offset register

REG_MATES_DAC5_MK1_COF06 : (1405) COF06 (#0x57D, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - OUT06 calibration offset register

REG_MATES_DAC5_MK1_COF07 : (1406) COF07 (#0x57E, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT07 calibration offset register

REG_MATES_DAC5_MK1_COF08 : (1407) COF08 (#0x57F, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT08 calibration offset register

REG_MATES_DAC5_MK1_COF09 : (1408) COF09 (#0x580, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT09 calibration offset register

REG_MATES_DAC5_MK1_COF10 : (1409) COF10 (#0x581, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT10 calibration offset register

REG_MATES_DAC5_MK1_COF11 : (1410) COF11 (#0x582, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT11 calibration offset register

REG_MATES_DAC5_MK1_COF12 : (1411) COF12 (#0x583, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT12 calibration offset register

REG_MATES_DAC5_MK1_COF13 : (1412) COF13 (#0x584, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT13 calibration offset register

REG_MATES_DAC5_MK1_COF14 : (1413) COF14 (#0x585, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT14 calibration offset register

REG_MATES_DAC5_MK1_COF15 : (1414) COF15 (#0x586, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT15 calibration offset register

REG_MATES_DAC5_MK1_COF16 : (1415) COF16 (#0x587, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT16 calibration offset register

REG_MATES_DAC5_MK1_COF17 : (1416) COF17 (#0x588, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT17 calibration offset register

REG_MATES_DAC5_MK1_COF18 : (1417) COF18 (#0x589, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT18 calibration offset register

REG_MATES_DAC5_MK1_COF19 : (1418) COF19 (#0x58A, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT19 calibration offset register

REG_MATES_DAC5_MK1_COF20 : (1419) COF20 (#0x58B, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT20 calibration offset register

REG_MATES_DAC5_MK1_COF21 : (1420) COF21 (#0x58C, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT21 calibration offset register

REG_MATES_DAC5_MK1_COF22 : (1421) COF22 (#0x58D, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT22 calibration offset register

REG_MATES_DAC5_MK1_COF23 : (1422) COF23 (#0x58E, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT23 calibration offset register

REG_MATES_DAC5_MK1_COF24 : (1423) COF24 (#0x58F, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT24 calibration offset register

REG_MATES_DAC5_MK1_COF25 : (1424) COF25 (#0x590, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT25 calibration offset register

REG_MATES_DAC5_MK1_COF26 : (1425) COF26 (#0x591, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT26 calibration offset register

REG_MATES_DAC5_MK1_COF27 : (1426) COF27 (#0x592, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT27 calibration offset register

REG_MATES_DAC5_MK1_COF28 : (1427) COF28 (#0x593, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT28 calibration offset register

REG_MATES_DAC5_MK1_COF29 : (1428) COF29 (#0x594, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT29 calibration offset register

REG_MATES_DAC5_MK1_COF30 : (1429) COF30 (#0x595, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT30 calibration offset register

REG_MATES_DAC5_MK1_COF31 : (1430) COF31 (#0x596, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT31 calibration offset register

REG_MATES_DAC5_MK1_COF32 : (1431) COF32 (#0x597, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT32 calibration offset register

REG_MATES_DAC5_MK1_COF33 : (1432) COF33 (#0x598, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT33 calibration offset register

REG_MATES_DAC5_MK1_COF34 : (1433) COF34 (#0x599, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT34 calibration offset register

REG_MATES_DAC5_MK1_COF35 : (1434) COF35 (#0x59A, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT35 calibration offset register

REG_MATES_DAC5_MK1_COF36 : (1435) COF36 (#0x59B, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT36 calibration offset register

REG_MATES_DAC5_MK1_COF37 : (1436) COF37 (#0x59C, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT37 calibration offset register

REG_MATES_DAC5_MK1_COF38 : (1437) COF38 (#0x59D, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT38 calibration offset register

REG_MATES_DAC5_MK1_COF39 : (1438) COF39 (#0x59E, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT39 calibration offset register

REG_MATES_DAC5_MK1_COF40 : (1439) COF40 (#0x59F, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - OUT40 calibration offset register

REG_MATES_DAC5_MK1_CGA01 : (1440) CGA01 (#0x5A0, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT01 calibration gain register

REG_MATES_DAC5_MK1_CGA02 : (1441) CGA02 (#0x5A1, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT02 calibration gain register

REG_MATES_DAC5_MK1_CGA03 : (1442) CGA03 (#0x5A2, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT03 calibration gain register

REG_MATES_DAC5_MK1_CGA04 : (1443) CGA04 (#0x5A3, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT04 calibration gain register

REG_MATES_DAC5_MK1_CGA05 : (1444) CGA05 (#0x5A4, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT05 calibration gain register

REG_MATES_DAC5_MK1_CGA06 : (1445) CGA06 (#0x5A5, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT06 calibration gain register

REG_MATES_DAC5_MK1_CGA07 : (1446) CGA07 (#0x5A6, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT07 calibration gain register

REG_MATES_DAC5_MK1_CGA08 : (1447) CGA08 (#0x5A7, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT08 calibration gain register

REG_MATES_DAC5_MK1_CGA09 : (1448) CGA09 (#0x5A8, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT09 calibration gain register

REG_MATES_DAC5_MK1_CGA10 : (1449) CGA10 (#0x5A9, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT10 calibration gain register

REG_MATES_DAC5_MK1_CGA11 : (1450) CGA11 (#0x5AA, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT11 calibration gain register

REG_MATES_DAC5_MK1_CGA12 : (1451) CGA12 (#0x5AB, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT12 calibration gain register

REG_MATES_DAC5_MK1_CGA13 : (1452) CGA13 (#0x5AC, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT13 calibration gain register

REG_MATES_DAC5_MK1_CGA14 : (1453) CGA14 (#0x5AD, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT14 calibration gain register

REG_MATES_DAC5_MK1_CGA15 : (1454) CGA15 (#0x5AE, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT15 calibration gain register

REG_MATES_DAC5_MK1_CGA16 : (1455) CGA16 (#0x5AF, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT16 calibration gain register

REG_MATES_DAC5_MK1_CGA17 : (1456) CGA17 (#0x5B0, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT17 calibration gain register

REG_MATES_DAC5_MK1_CGA18 : (1457) CGA18 (#0x5B1, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT18 calibration gain register

REG_MATES_DAC5_MK1_CGA19 : (1458) CGA19 (#0x5B2, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT19 calibration gain register

REG_MATES_DAC5_MK1_CGA20 : (1459) CGA20 (#0x5B3, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT20 calibration gain register

REG_MATES_DAC5_MK1_CGA21 : (1460) CGA21 (#0x5B4, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT21 calibration gain register

REG_MATES_DAC5_MK1_CGA22 : (1461) CGA22 (#0x5B5, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT22 calibration gain register

REG_MATES_DAC5_MK1_CGA23 : (1462) CGA23 (#0x5B6, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT23 calibration gain register

REG_MATES_DAC5_MK1_CGA24 : (1463) CGA24 (#0x5B7, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT24 calibration gain register

REG_MATES_DAC5_MK1_CGA25 : (1464) CGA25 (#0x5B8, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT25 calibration gain register

REG_MATES_DAC5_MK1_CGA26 : (1465) CGA26 (#0x5B9, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT26 calibration gain register

REG_MATES_DAC5_MK1_CGA27 : (1466) CGA27 (#0x5BA, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT27 calibration gain register

REG_MATES_DAC5_MK1_CGA28 : (1467) CGA28 (#0x5BB, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT28 calibration gain register

REG_MATES_DAC5_MK1_CGA29 : (1468) CGA29 (#0x5BC, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT29 calibration gain register

REG_MATES_DAC5_MK1_CGA30 : (1469) CGA30 (#0x5BD, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT30 calibration gain register

REG_MATES_DAC5_MK1_CGA31 : (1470) CGA31 (#0x5BE, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT31 calibration gain register

REG_MATES_DAC5_MK1_CGA32 : (1471) CGA32 (#0x5BF, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT32 calibration gain register

REG_MATES_DAC5_MK1_CGA33 : (1472) CGA33 (#0x5C0, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT33 calibration gain register

REG_MATES_DAC5_MK1_CGA34 : (1473) CGA34 (#0x5C1, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT34 calibration gain register

REG_MATES_DAC5_MK1_CGA35 : (1474) CGA35 (#0x5C2, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT35 calibration gain register

REG_MATES_DAC5_MK1_CGA36 : (1475) CGA36 (#0x5C3, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT36 calibration gain register

REG_MATES_DAC5_MK1_CGA37 : (1476) CGA37 (#0x5C4, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT37 calibration gain register

REG_MATES_DAC5_MK1_CGA38 : (1477) CGA38 (#0x5C5, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT38 calibration gain register

REG_MATES_DAC5_MK1_CGA39 : (1478) CGA39 (#0x5C6, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT39 calibration gain register

REG_MATES_DAC5_MK1_CGA40 : (1479) CGA40 (#0x5C7, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - OUT40 calibration gain register

REG_MATES_DAC5_MK1_BMIN : (1480) BMIN (#0x5C8, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFF) - Minimum binary DAC code for all channels register

REG_MATES_DAC5_MK1_BMAX : (1481) BMAX (#0x5C9, RWN, init.: 0x9A00, min.: 0x0, max.: 0xFFFF) - Maximum binary DAC code for all channels register

REG_MATES_DAC5_MK1_LCT : (1482) LCT (#0x5CA, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Last calibration time register

REG_MATES_DAC5_MK1_CANID : (1483) CANID (#0x5CB, RWN, init.: 50, min.: 50, max.: 59) - The node's CAN ID / address register

REG_MATES_DIOX_MK1_ODEF : (1024) ODEF (#0x400, RWN, init.: 0xC0000000, min.: 0x0, max.: 0xC00FFFFF) - Outputs default value register.

REG_MATES_DIOX_MK1_OCTRL01 : (1025) OCTRL01 (#0x401, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT01 control register

REG_MATES_DIOX_MK1_OCTRL02 : (1026) OCTRL02 (#0x402, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT02 control register

REG_MATES_DIOX_MK1_OCTRL03 : (1027) OCTRL03 (#0x403, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT03 control register

REG_MATES_DIOX_MK1_OCTRL04 : (1028) OCTRL04 (#0x404, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT04 control register

REG_MATES_DIOX_MK1_OCTRL05 : (1029) OCTRL05 (#0x405, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT05 control register

REG_MATES_DIOX_MK1_OCTRL06 : (1030) OCTRL06 (#0x406, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT06 control register

REG_MATES_DIOX_MK1_OCTRL07 : (1031) OCTRL07 (#0x407, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT07 control register

REG_MATES_DIOX_MK1_OCTRL08 : (1032) OCTRL08 (#0x408, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT08 control register

REG_MATES_DIOX_MK1_OCTRL09 : (1033) OCTRL09 (#0x409, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT09 control register

REG_MATES_DIOX_MK1_OCTRL10 : (1034) OCTRL10 (#0x40A, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT10 control register

REG_MATES_DIOX_MK1_OCTRL11 : (1035) OCTRL11 (#0x40B, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT11 control register

REG_MATES_DIOX_MK1_OCTRL12 : (1036) OCTRL12 (#0x40C, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT12 control register

REG_MATES_DIOX_MK1_OCTRL13 : (1037) OCTRL13 (#0x40D, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT13 control register

REG_MATES_DIOX_MK1_OCTRL14 : (1038) OCTRL14 (#0x40E, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT14 control register

REG_MATES_DIOX_MK1_OCTRL15 : (1039) OCTRL15 (#0x40F, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT15 control register

REG_MATES_DIOX_MK1_OCTRL16 : (1040) OCTRL16 (#0x410, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT16 control register

REG_MATES_DIOX_MK1_OCTRL17 : (1041) OCTRL17 (#0x411, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT17 control register

REG_MATES_DIOX_MK1_OCTRL18 : (1042) OCTRL18 (#0x412, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT18 control register

REG_MATES_DIOX_MK1_OCTRL19 : (1043) OCTRL19 (#0x413, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT19 control register

REG_MATES_DIOX_MK1_OCTRL20 : (1044) OCTRL20 (#0x414, RWN, init.: 0x0, min.: 0x0, max.: 0x7) - OUT20 control register

REG_MATES_DIOX_MK1_ORP : (1045) ORP (#0x415, RWN, init.: 0x0, min.: 0x0, max.: 0x↩FFFFFFFF) - Outputs restore period register

REG_MATES_DIOX_MK1_ORRT : (1046) ORRT (#0x416, RO, init.: 0x0) - Outptus restore remaining time register

REG_MATES_DIOX_MK1_CANID : (1047) CANID (#0x417, RWN, init.: 40, min.: 40, max.: 49) - The node's CAN ID / address register

REG_MATES_ADC5_MK1_COF01 : (1536) COF01 (#0x600, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN01 calibration offset register

REG_MATES_ADC5_MK1_COF02 : (1537) COF02 (#0x601, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN02 calibration offset register

REG_MATES_ADC5_MK1_COF03 : (1538) COF03 (#0x602, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN03 calibration offset register

REG_MATES_ADC5_MK1_COF04 : (1539) COF04 (#0x603, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN04 calibration offset register

REG_MATES_ADC5_MK1_COF05 : (1540) COF05 (#0x604, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN05 calibration offset register

REG_MATES_ADC5_MK1_COF06 : (1541) COF06 (#0x605, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN06 calibration offset register

REG_MATES_ADC5_MK1_COF07 : (1542) COF07 (#0x606, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN07 calibration offset register

REG_MATES_ADC5_MK1_COF08 : (1543) COF08 (#0x607, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN08 calibration offset register

REG_MATES_ADC5_MK1_COF09 : (1544) COF09 (#0x608, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN09 calibration offset register

REG_MATES_ADC5_MK1_COF10 : (1545) COF10 (#0x609, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN10 calibration offset register

REG_MATES_ADC5_MK1_COF11 : (1546) COF11 (#0x60A, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - IN11 calibration offset register

REG_MATES_ADC5_MK1_COF12 : (1547) COF12 (#0x60B, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - IN12 calibration offset register

REG_MATES_ADC5_MK1_COF13 : (1548) COF13 (#0x60C, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - IN13 calibration offset register

REG_MATES_ADC5_MK1_COF14 : (1549) COF14 (#0x60D, RWN, init.: 0.0, min.: -0.1, max.↩ : 0.1) - IN14 calibration offset register

REG_MATES_ADC5_MK1_COF15 : (1550) COF15 (#0x60E, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN15 calibration offset register

REG_MATES_ADC5_MK1_COF16 : (1551) COF16 (#0x60F, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN16 calibration offset register

REG_MATES_ADC5_MK1_COF17 : (1552) COF17 (#0x610, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN17 calibration offset register

REG_MATES_ADC5_MK1_COF18 : (1553) COF18 (#0x611, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN18 calibration offset register

REG_MATES_ADC5_MK1_COF19 : (1554) COF19 (#0x612, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN19 calibration offset register

REG_MATES_ADC5_MK1_COF20 : (1555) COF20 (#0x613, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN20 calibration offset register

REG_MATES_ADC5_MK1_COF21 : (1556) COF21 (#0x614, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN21 calibration offset register

REG_MATES_ADC5_MK1_COF22 : (1557) COF22 (#0x615, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN22 calibration offset register

REG_MATES_ADC5_MK1_COF23 : (1558) COF23 (#0x616, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN23 calibration offset register

REG_MATES_ADC5_MK1_COF24 : (1559) COF24 (#0x617, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN24 calibration offset register

REG_MATES_ADC5_MK1_COF25 : (1560) COF25 (#0x618, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN25 calibration offset register

REG_MATES_ADC5_MK1_COF26 : (1561) COF26 (#0x619, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN26 calibration offset register

REG_MATES_ADC5_MK1_COF27 : (1562) COF27 (#0x61A, RWN, init.: 0.0, min.: -0.1, max.↩: 0.1) - IN27 calibration offset register

REG_MATES_ADC5_MK1_COF28 : (1563) COF28 (#0x61B, RWN, init.: 0.0, min.: -0.1, max.↩: 0.1) - IN28 calibration offset register

REG_MATES_ADC5_MK1_COF29 : (1564) COF29 (#0x61C, RWN, init.: 0.0, min.: -0.1, max.↩: 0.1) - IN29 calibration offset register

REG_MATES_ADC5_MK1_COF30 : (1565) COF30 (#0x61D, RWN, init.: 0.0, min.: -0.1, max.↩: 0.1) - IN30 calibration offset register

REG_MATES_ADC5_MK1_COF31 : (1566) COF31 (#0x61E, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN31 calibration offset register

REG_MATES_ADC5_MK1_COF32 : (1567) COF32 (#0x61F, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN32 calibration offset register

REG_MATES_ADC5_MK1_COF33 : (1568) COF33 (#0x620, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN33 calibration offset register

REG_MATES_ADC5_MK1_COF34 : (1569) COF34 (#0x621, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN34 calibration offset register

REG_MATES_ADC5_MK1_COF35 : (1570) COF35 (#0x622, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN35 calibration offset register

REG_MATES_ADC5_MK1_COF36 : (1571) COF36 (#0x623, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN36 calibration offset register

REG_MATES_ADC5_MK1_COF37 : (1572) COF37 (#0x624, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN37 calibration offset register

REG_MATES_ADC5_MK1_COF38 : (1573) COF38 (#0x625, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN38 calibration offset register

REG_MATES_ADC5_MK1_COF39 : (1574) COF39 (#0x626, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN39 calibration offset register

REG_MATES_ADC5_MK1_COF40 : (1575) COF40 (#0x627, RWN, init.: 0.0, min.: -0.1, max.: 0.1) - IN40 calibration offset register

REG_MATES_ADC5_MK1_CGA01 : (1576) CGA01 (#0x628, RWN, init.: 1.0, min.: 0.95, max.↩: 1.05) - IN01 calibration gain register

REG_MATES_ADC5_MK1_CGA02 : (1577) CGA02 (#0x629, RWN, init.: 1.0, min.: 0.95, max.↩: 1.05) - IN02 calibration gain register

REG_MATES_ADC5_MK1_CGA03 : (1578) CGA03 (#0x62A, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN03 calibration gain register

REG_MATES_ADC5_MK1_CGA04 : (1579) CGA04 (#0x62B, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN04 calibration gain register

REG_MATES_ADC5_MK1_CGA05 : (1580) CGA05 (#0x62C, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN05 calibration gain register

REG_MATES_ADC5_MK1_CGA06 : (1581) CGA06 (#0x62D, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN06 calibration gain register

REG_MATES_ADC5_MK1_CGA07 : (1582) CGA07 (#0x62E, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN07 calibration gain register

REG_MATES_ADC5_MK1_CGA08 : (1583) CGA08 (#0x62F, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN08 calibration gain register

REG_MATES_ADC5_MK1_CGA09 : (1584) CGA09 (#0x630, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN09 calibration gain register

REG_MATES_ADC5_MK1_CGA10 : (1585) CGA10 (#0x631, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN10 calibration gain register

REG_MATES_ADC5_MK1_CGA11 : (1586) CGA11 (#0x632, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN11 calibration gain register

REG_MATES_ADC5_MK1_CGA12 : (1587) CGA12 (#0x633, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN12 calibration gain register

REG_MATES_ADC5_MK1_CGA13 : (1588) CGA13 (#0x634, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN13 calibration gain register

REG_MATES_ADC5_MK1_CGA14 : (1589) CGA14 (#0x635, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN14 calibration gain register

REG_MATES_ADC5_MK1_CGA15 : (1590) CGA15 (#0x636, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN15 calibration gain register

REG_MATES_ADC5_MK1_CGA16 : (1591) CGA16 (#0x637, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN16 calibration gain register

REG_MATES_ADC5_MK1_CGA17 : (1592) CGA17 (#0x638, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN17 calibration gain register

REG_MATES_ADC5_MK1_CGA18 : (1593) CGA18 (#0x639, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN18 calibration gain register

REG_MATES_ADC5_MK1_CGA19 : (1594) CGA19 (#0x63A, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN19 calibration gain register

REG_MATES_ADC5_MK1_CGA20 : (1595) CGA20 (#0x63B, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN20 calibration gain register

REG_MATES_ADC5_MK1_CGA21 : (1596) CGA21 (#0x63C, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN21 calibration gain register

REG_MATES_ADC5_MK1_CGA22 : (1597) CGA22 (#0x63D, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN22 calibration gain register

REG_MATES_ADC5_MK1_CGA23 : (1598) CGA23 (#0x63E, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN23 calibration gain register

REG_MATES_ADC5_MK1_CGA24 : (1599) CGA24 (#0x63F, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN24 calibration gain register

REG_MATES_ADC5_MK1_CGA25 : (1600) CGA25 (#0x640, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN25 calibration gain register

REG_MATES_ADC5_MK1_CGA26 : (1601) CGA26 (#0x641, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN26 calibration gain register

REG_MATES_ADC5_MK1_CGA27 : (1602) CGA27 (#0x642, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN27 calibration gain register

REG_MATES_ADC5_MK1_CGA28 : (1603) CGA28 (#0x643, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN28 calibration gain register

REG_MATES_ADC5_MK1_CGA29 : (1604) CGA29 (#0x644, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN29 calibration gain register

REG_MATES_ADC5_MK1_CGA30 : (1605) CGA30 (#0x645, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN30 calibration gain register

REG_MATES_ADC5_MK1_CGA31 : (1606) CGA31 (#0x646, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN31 calibration gain register

REG_MATES_ADC5_MK1_CGA32 : (1607) CGA32 (#0x647, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN32 calibration gain register

REG_MATES_ADC5_MK1_CGA33 : (1608) CGA33 (#0x648, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN33 calibration gain register

REG_MATES_ADC5_MK1_CGA34 : (1609) CGA34 (#0x649, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN34 calibration gain register

REG_MATES_ADC5_MK1_CGA35 : (1610) CGA35 (#0x64A, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN35 calibration gain register

REG_MATES_ADC5_MK1_CGA36 : (1611) CGA36 (#0x64B, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN36 calibration gain register

REG_MATES_ADC5_MK1_CGA37 : (1612) CGA37 (#0x64C, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN37 calibration gain register

REG_MATES_ADC5_MK1_CGA38 : (1613) CGA38 (#0x64D, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN38 calibration gain register

REG_MATES_ADC5_MK1_CGA39 : (1614) CGA39 (#0x64E, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN39 calibration gain register

REG_MATES_ADC5_MK1_CGA40 : (1615) CGA40 (#0x64F, RWN, init.: 1.0, min.: 0.95, max.↩ : 1.05) - IN40 calibration gain register

REG_MATES_ADC5_MK1_LCT : (1616) LCT (#0x650, RWN, init.: 0x0, min.: 0x0, max.: 0x↩ FFFFFFFF) - Last calibration time register

REG_MATES_ADC5_MK1_CANID : (1617) CANID (#0x651, RWN, init.: 60, min.: 60, max.: 69) - The node's CAN ID / address register

REG_MATES_UCC_MK1_ODEF : (768) ODEF (#0x300, RWN, init.: 0x0, min.: 0x0, max.: 0xFFFFF) - Outputs default value register

REG_MATES_UCC_MK1_ODIS : (769) ODIS (#0x301, RW, init.: 0x0, min.: 0x0, max.: 0xF↩ FFFF) - Outputs disable register

REG_MATES_UCC_MK1_PRD01 : (770) PRD01 (#0x302, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 01 period register

REG_MATES_UCC_MK1_PRD02 : (771) PRD02 (#0x303, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 02 period register

REG_MATES_UCC_MK1_PRD03 : (772) PRD03 (#0x304, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 03 period register

REG_MATES_UCC_MK1_PRD04 : (773) PRD04 (#0x305, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 04 period register

REG_MATES_UCC_MK1_PRD05 : (774) PRD05 (#0x306, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 05 period register

REG_MATES_UCC_MK1_PRD06 : (775) PRD06 (#0x307, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 06 period register

REG_MATES_UCC_MK1_PRD07 : (776) PRD07 (#0x308, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 07 period register

REG_MATES_UCC_MK1_PRD08 : (777) PRD08 (#0x309, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 08 period register

REG_MATES_UCC_MK1_PRD09 : (778) PRD09 (#0x30A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 09 period register

REG_MATES_UCC_MK1_PRD10 : (779) PRD10 (#0x30B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 10 period register

REG_MATES_UCC_MK1_PRD11 : (780) PRD11 (#0x30C, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 11 period register

REG_MATES_UCC_MK1_PRD12 : (781) PRD12 (#0x30D, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 12 period register

REG_MATES_UCC_MK1_PRD13 : (782) PRD13 (#0x30E, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 13 (K1) period register

REG_MATES_UCC_MK1_PRD14 : (783) PRD14 (#0x30F, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 14 (K2) period register

REG_MATES_UCC_MK1_TRISE01 : (784) TRISE01 (#0x310, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 01 rise time register

REG_MATES_UCC_MK1_TRISE02 : (785) TRISE02 (#0x311, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 02 rise time register

REG_MATES_UCC_MK1_TRISE03 : (786) TRISE03 (#0x312, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 03 rise time register

REG_MATES_UCC_MK1_TRISE04 : (787) TRISE04 (#0x313, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 04 rise time register

REG_MATES_UCC_MK1_TRISE05 : (788) TRISE05 (#0x314, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 05 rise time register

REG_MATES_UCC_MK1_TRISE06 : (789) TRISE06 (#0x315, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 06 rise time register

REG_MATES_UCC_MK1_TRISE07 : (790) TRISE07 (#0x316, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 07 rise time register

REG_MATES_UCC_MK1_TRISE08 : (791) TRISE08 (#0x317, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 08 rise time register

REG_MATES_UCC_MK1_TRISE09 : (792) TRISE09 (#0x318, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 09 rise time register

REG_MATES_UCC_MK1_TRISE10 : (793) TRISE10 (#0x319, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 10 rise time register

REG_MATES_UCC_MK1_TRISE11 : (794) TRISE11 (#0x31A, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 11 rise time register

REG_MATES_UCC_MK1_TRISE12 : (795) TRISE12 (#0x31B, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 12 rise time register

REG_MATES_UCC_MK1_TRISE13 : (796) TRISE13 (#0x31C, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 13 (K1) rise time register

REG_MATES_UCC_MK1_TRISE14 : (797) TRISE14 (#0x31D, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 14 (K2) rise time register

REG_MATES_UCC_MK1_TFALL01 : (798) TFALL01 (#0x31E, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 01 fall time register

REG_MATES_UCC_MK1_TFALL02 : (799) TFALL02 (#0x31F, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 02 fall time register

REG_MATES_UCC_MK1_TFALL03 : (800) TFALL03 (#0x320, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 03 fall time register

REG_MATES_UCC_MK1_TFALL04 : (801) TFALL04 (#0x321, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 04 fall time register

REG_MATES_UCC_MK1_TFALL05 : (802) TFALL05 (#0x322, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 05 fall time register

REG_MATES_UCC_MK1_TFALL06 : (803) TFALL06 (#0x323, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 06 fall time register

REG_MATES_UCC_MK1_TFALL07 : (804) TFALL07 (#0x324, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 07 fall time register

REG_MATES_UCC_MK1_TFALL08 : (805) TFALL08 (#0x325, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 08 fall time register

REG_MATES_UCC_MK1_TFALL09 : (806) TFALL09 (#0x326, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 09 fall time register

REG_MATES_UCC_MK1_TFALL10 : (807) TFALL10 (#0x327, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 10 fall time register

REG_MATES_UCC_MK1_TFALL11 : (808) TFALL11 (#0x328, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 11 fall time register

REG_MATES_UCC_MK1_TFALL12 : (809) TFALL12 (#0x329, RW, init.: 0x0, min.: 0x0, max.↩ : 0xFFFFFFFF) - Channel 12 fall time register

REG_MATES_UCC_MK1_TFALL13 : (810) TFALL13 (#0x32A, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 13 (K1) fall time register

REG_MATES_UCC_MK1_TFALL14 : (811) TFALL14 (#0x32B, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Channel 14 (K2) fall time register

REG_MATES_UCC_MK1_ORP : (812) ORP (#0x32C, RWN, init.: 0x0, min.: 0x0, max.: 0x↩ FFFFFFFF) - Outputs restore period

REG_MATES_UCC_MK1_ORRT : (813) ORRT (#0x32D, RO, init.: 0x0) - Outptus restore remaining time register

REG_MATES_UCC_MK1_CANID : (814) CANID (#0x32E, RWN, init.: 30, min.: 30, max.: 39) - The node's CAN ID / address register

REG_MATES_UCC_MK1_PWMD : (815) PWMD (#0x32F, RW, init.: 0.0, min.: 0.0, max.: 1.0) - Pulse Width Modulator Duty

REG_MATES_UCC_MK1_ALCDB01 : (816) ALCDB01 (#0x330, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 1

REG_MATES_UCC_MK1_ALCDB02 : (817) ALCDB02 (#0x331, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 2

REG_MATES_UCC_MK1_ALCDB03 : (818) ALCDB03 (#0x332, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 3

REG_MATES_UCC_MK1_ALCDB04 : (819) ALCDB04 (#0x333, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 4

REG_MATES_UCC_MK1_ALCDB05 : (820) ALCDB05 (#0x334, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 5

REG_MATES_UCC_MK1_ALCDB06 : (821) ALCDB06 (#0x335, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 6

REG_MATES_UCC_MK1_ALCDB07 : (822) ALCDB07 (#0x336, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 7

REG_MATES_UCC_MK1_ALCDB08 : (823) ALCDB08 (#0x337, RW, init.: 0x0, min.: 0x0, max.: 0xFFFFFFFF) - Alphanumeric LCD display data register 8

B List of available VISA queries

This section lists the VISA queries that are supported by the devices. Unless otherwise specified, the queries can be executed on all MATES devices.

All queries available on MATES devices can be divided into 3 basic groups:

1. Read - only queries (such as `*IDN?`).
2. Commands (such as `*RST`).
3. Read - write queries (such as `P:ENABLE`).

The devices provide queries defined below.

B.1 Queries by function

B.1.1 MATES status and control

- *IDN?
- *RST
- *TST?
- C:TUT?
- I:BRD?
- R:RENA and R:RST
- S:STA?

B.1.2 MATES parameters

- P:ENABLE
- P:CID

B.2 Read - only queries

B.2.1 *IDN?

Returns the device identification string:

```
1 VIRESCO,MATES-DIO3-MK1,0001,1.1.526.0
```

B.2.2 *TST?

Returns current BIT status. The status is presented by a number followed by a short description. Number of 0 denotes no - error condition.

→ If there are more than one BIT failures present, they will be presented in a cycle, each fault for ∼2 seconds.

```
1 *TST? -> 2 UMAINS HI
2 *TST? -> 0 OK
```

B.2.3 C:TUT?

Returns total operating time in seconds:

```
C:TUT? -> 385601 s
```

B.2.4 I:BRD?

Returns board identification information:

```
I:BRD? -> SN 274 BA 7 BI 4 CI 0
```

B.2.5 S:STA?

Returns current operating state:

```
S:STA? -> 0 ERROR
S:STA? -> 1 START
```

B.3 Commands

B.3.1 *RST

Returns all parameters to their default values and clear any BIT faults:

```
*RST -> VI: reset OK
```

B.3.2 R:RENA and R:RST

These two commands are used to reset the device. They have to be executed one after another with no more than 3 seconds of separation. Note that you won't receive response for the second query (it will time-out).

```
R:RENA ->  R:RENA OK
R:RST -> (no response)
```

B.4 Read - write queries

B.4.1 P:ENABLE

Defines the various configuration flags:

- bit 0 [cbit] (LSB) - enable BIT;
- bit 1 [bod] - enable BOD (brown out detector);

```
P:ENABLE? -> 0x3
P:ENABLE 0 -> VI: Enable = 0x0
P:ENABLE DEF -> VI: Enable = 0x3
```

B.4.2 P:CID

Defines the device CAN address:

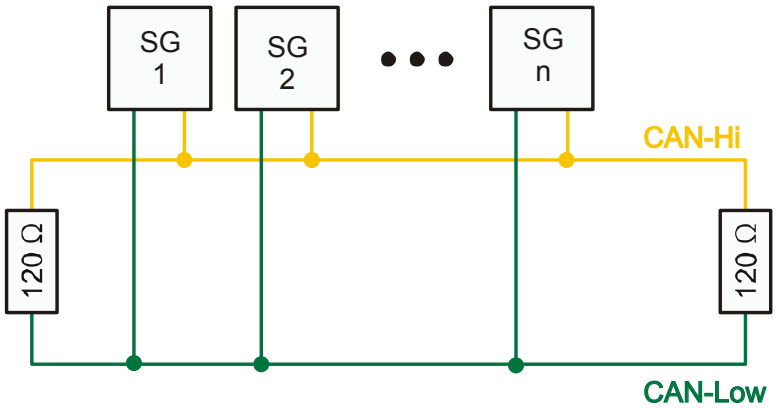
```
1 P:CID? -> 50
2 P:CID 52 -> VI: Cid = 52
3 P:CID? -> 52
```

C CAN Protocol

The CAN protocol[7] is an international standard defined in the ISO 11898 for high speed and ISO 11519-2 for low speed.

The CAN protocol used by MATES operates on the basic bus topology (see Figure 21).

Figure 21: CAN basic topology



Each CAN device is connected to a differential pair of wires. The bus is terminated at each end by the terminating resistors.

C.1 Principles

CAN is based on a broadcast communication mechanism. This broadcast communication is achieved by using a message oriented transmission protocol. These messages are identified by using a message identifier. Such a message identifier has to be unique within the whole network and it defines not only the content but also the priority of the message. The priority at which a message is transmitted compared to another less urgent message is specified by the identifier of each message. The priorities are laid down during system design in the form of corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority. Bus access conflicts are resolved by bit-wise arbitration on the identifiers involved by each node observing the bus level bit for bit. This happens in accordance with the "wired and" mechanism, by which the dominant state overwrites the recessive state. The competition for bus allocation is lost by all nodes with recessive transmission and dominant observation. All the "losers" automatically become receivers of the message with the highest priority and do not re-attempt transmission until the bus is available again.

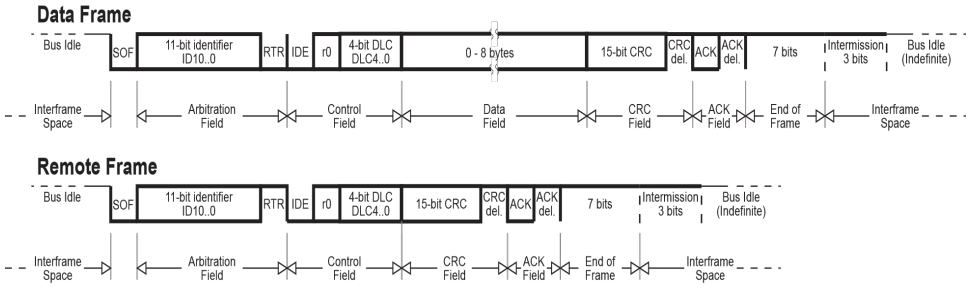
C.2 Message Formats

The CAN protocol supports two message frame formats, the only essential difference being in the length of the identifier. The CAN standard frame, also known as CAN 2.0 A, supports a length of 11 bits for the identifier, and the CAN extended frame, also known as CAN 2.0 B, supports a length of 29 bits for the identifier.

C.2.1 Can Standard Frame

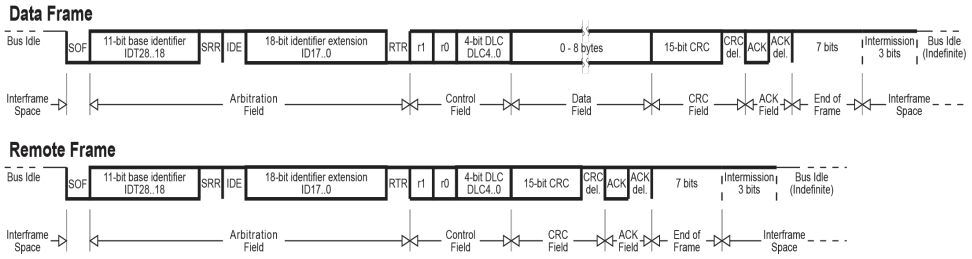
A message in the CAN standard frame format begins with the "Start Of Frame (SOF)", this is followed by the "Arbitration field" which consist of the identifier and the "Remote Transmission Request (RTR)" bit used to distinguish between the

[7] CAN protocol description is based on [1]



data frame and the data request frame called remote frame. The following "Control field" contains the "IDentifier Extension (IDE)" bit and the "Data Length Code (DLC)" used to indicate the number of following data bytes in the "Data field". In a remote frame, the DLC contains the number of requested data bytes. The "Data field" that follows can hold up to 8 data bytes. The frame integrity is guaranteed by the following "Cyclic Redundant Check (CRC)" sum. The "ACKnowledge (ACK) field" compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by the receivers which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by "End Of Frame (EOF)". The "Intermission Frame Space (IFS)" is the minimum number of bits separating consecutive messages. If there is no following bus access by any node, the bus remains idle.

C.2.2 CAN Extended Frame



A message in the CAN extended frame format is likely the same as a message in CAN standard frame format. The difference is the length of the identifier used. The identifier is made up of the existing 11-bit identifier (base identifier) and an 18-bit extension (identifier extension). The distinction between CAN standard frame format and CAN extended frame format is made by using the IDE bit which is transmitted as dominant in case of a frame in CAN standard frame format, and transmitted as recessive in the other case.

C.2.3 Format Co-existence

As the two formats have to co-exist on one bus, it is laid down which message has higher priority on the bus in the case of bus access collision with different formats and the same identifier / base identifier: The message in CAN standard frame format always has priority over the message in extended format. There are three different types of CAN modules available:

- 2.0A - Considers 29 bit ID as an error
- 2.0B Passive - Ignores 29 bit ID messages
- 2.0B Active - Handles both 11 and 29 bit ID Messages

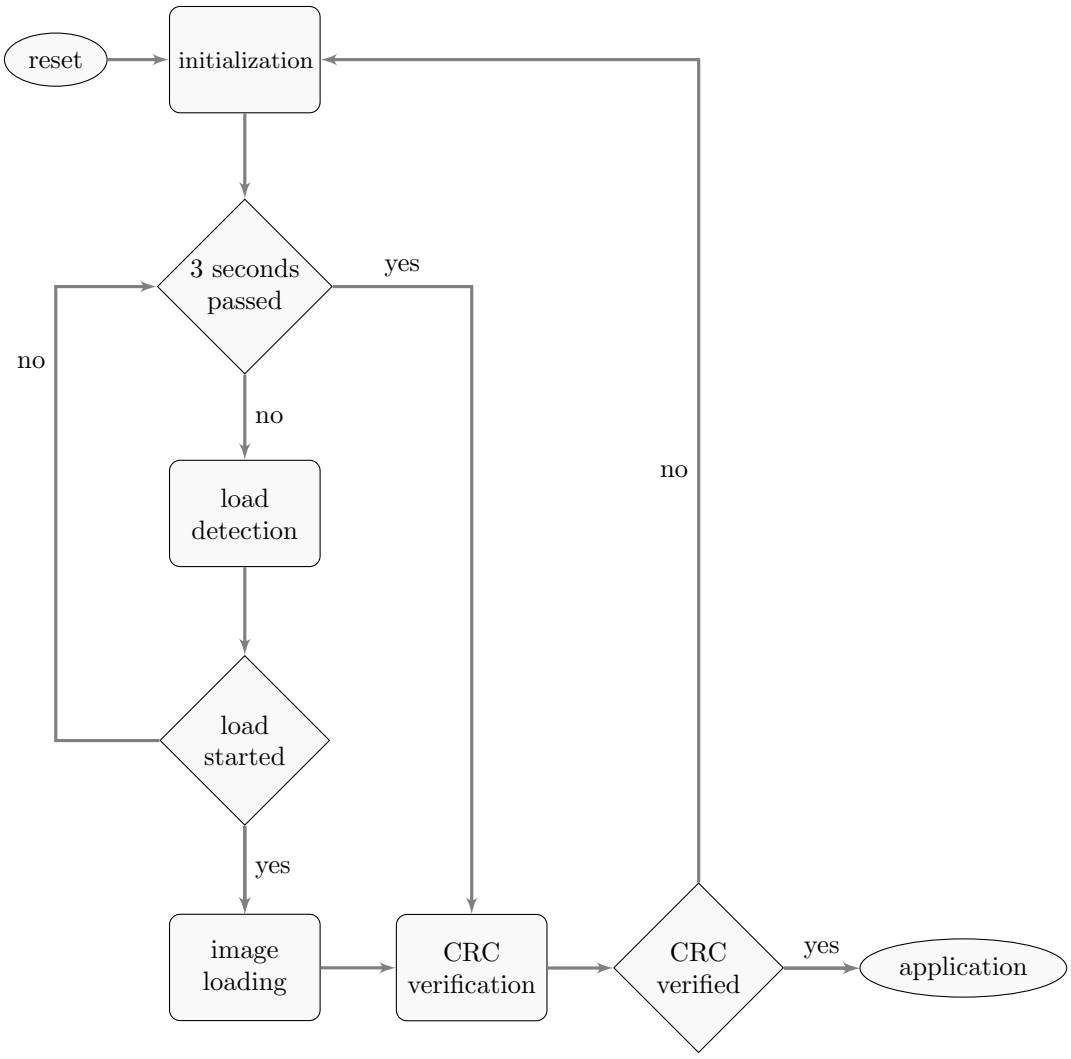
D Boot loader support

D.1 Starting boot loader program

In the current version (UCC-BL v2.4), the loading program implements the following algorithm:

1. After powering up the device, the loader program waits for the new image to be sent through RS-232 for the time of 3 seconds.
2. If the image transmission is initiated within the 3 seconds time and the loading program determines that the image is consistent, the image is loaded to the device's FLASH memory.
3. Whether the new image was loaded or not, the loading program verifies the consistency of the currently loaded image.
4. If the image is consistent, the main application is started, otherwise the loading program restarts.

Figure 22: Loading program flow



→ The bootloader program is automatically started during power up or after microcontroller reset. A watchdog reset can be used to indirectly start the bootloader program from the main application.

E MK1 nodes memory map

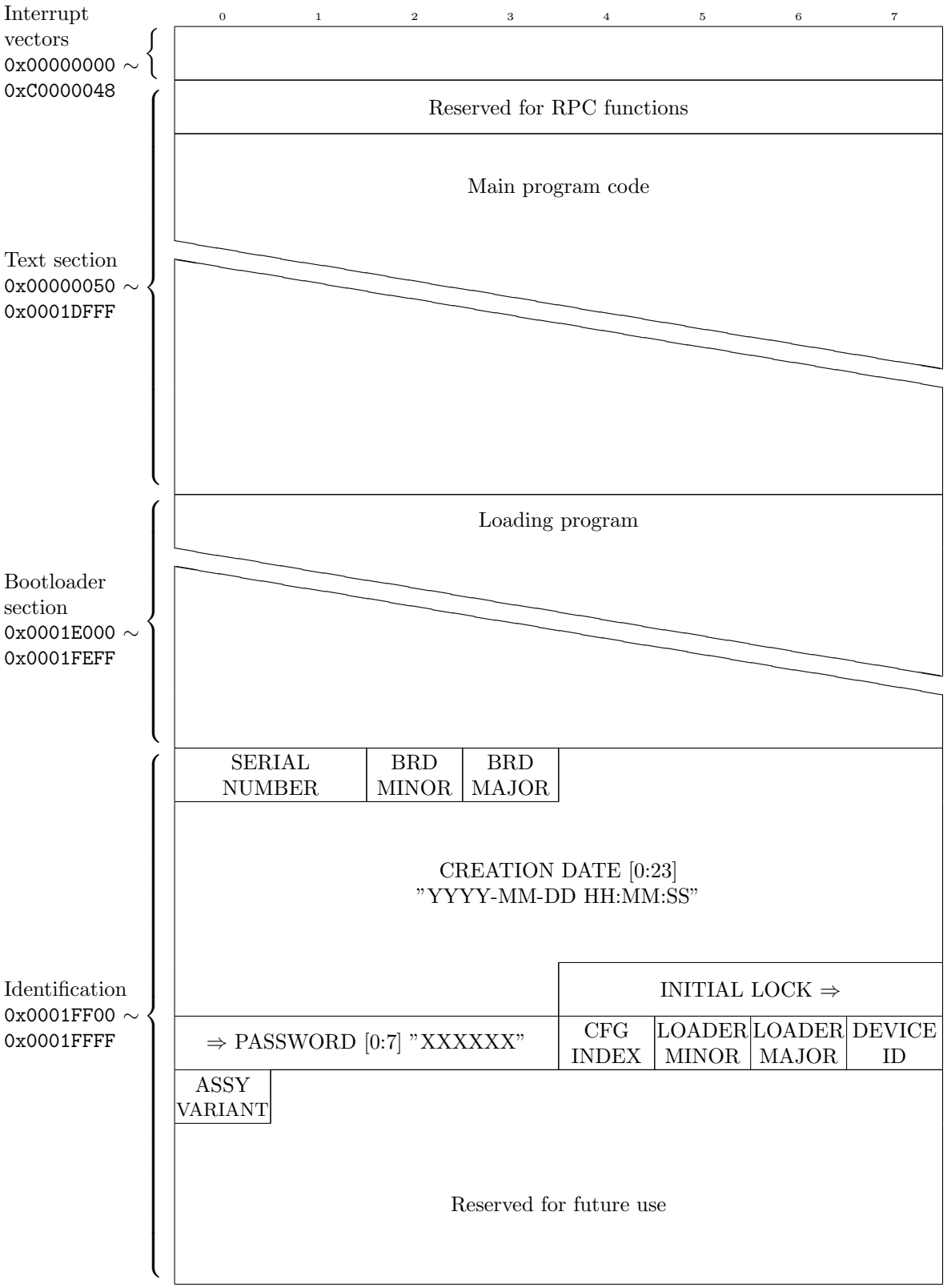


Figure 23: FLASH memory organization

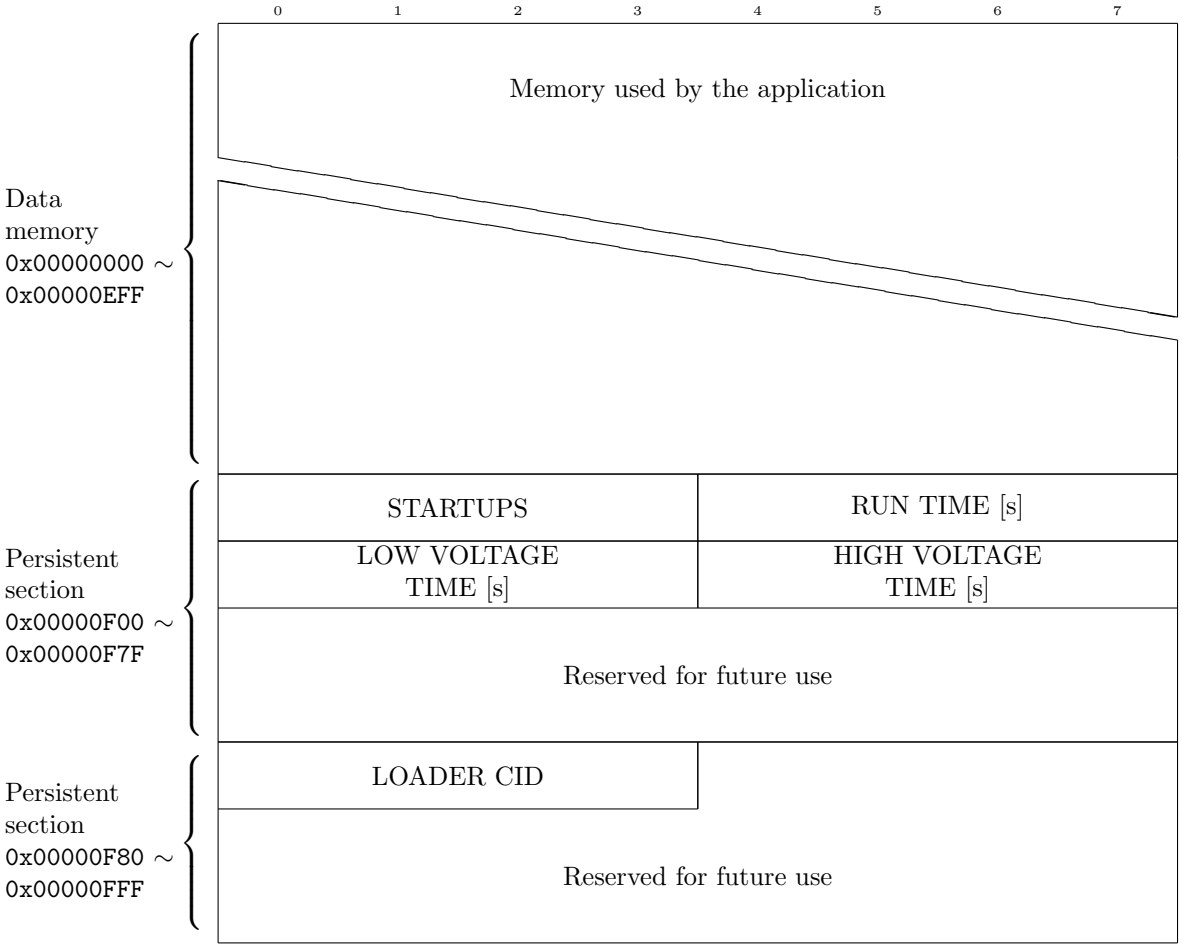


Figure 24: EEPROM memory organization

F Installation directory content

Note: some files where omitted for brevity.

```
/MATES ..... Installation root directory
├── api
│   ├── c
│   │   ├── mates.h ..... mates.dll header file
│   │   ├── mates_regs.h ..... MATES registers definitions header file
│   │   └── mates_types.h ..... MATES types definitions header file
│   ├── py
│   │   ├── mates
│   │   │   ├── __init__.py ..... Python wrapper package for mates.dll
│   │   │   └── regs.py ..... Python equivalent of mates_regs.h
│   │   ├── mates_console_default.bat ..... Startup script for mates_console.py
│   │   ├── mates_console_proxy.bat ..... Startup script for mates_console.py
│   │   └── mates_console.py ..... Convenient console for using mates.py
│   └── cs
│       ├── Adc5Node.cs ..... ADC5 node class
│       ├── Dac5Node.cs ..... DAC5 node class
│       ├── Dio3Node.cs ..... DIO3 node class
│       ├── DioNode.cs ..... DIO abstract base class
│       ├── DioxNode.cs ..... DIOX node class
│       ├── Mates.cs ..... MATES system main class
│       ├── MatesExceptions.cs ..... MATES system exceptions definitions
│       ├── MatesImports.cs ..... mates.dll wrapper
│       ├── MatesRegs.cs ..... C# equivalent of mates_regs.h
│       ├── MatesTypes.*.cs ..... C# register types definitions
│       ├── Node.cs ..... Base class for all types of nodes
│       └── TestStands.cs ..... Test stands implementations
├── bin
│   ├── default.mon ..... Default connection setting file (direct connection using COM1)
│   ├── mates.dll ..... The MATES system access library
│   ├── mates_proxy.exe ..... MATES access proxy server
│   ├── Viresco.Mates.dll ..... C# MATES class library
│   └── mates-*.whl ..... Python wheel packages
├── documents
│   ├── MATES.pdf ..... This document
│   ├── MATES_API.chm ..... MATES library API manual
│   └── uOS_monitor_EN.pdf ..... uOS_monitor program manual
├── firmware
│   ├── MATES_loader_*.exe ..... MATES loader tool package
│   └── mates_*.exe ..... Individual nodes firmware update packages
├── tests
│   ├── bin
│   │   └── mates_test_*.exe ..... Binary tests compiled from mates_test_*.c
│   ├── c
│   │   └── mates_test_*.c ..... Tests source files (also used as examples in API documentation)
│   ├── py
│   │   └── mates_test_*.py ..... Python test files (also used as examples in API documentation)
│   ├── cs
│   │   └── mates_test_*.cs ..... C# test files (also used as examples in API documentation)
│   ├── proxy.mon ..... Connection setting file when using MATES proxy
│   └── runall.py ..... Test suite main file
└── uninstall.exe ..... Uninstaller program
```

G Revision history

Version	Date	Change descirpion
60$^{(r4083)}$	*2019-10-17*	Fixed broken links [resolves #185].
59$^{(r3663)}$	*2018-03-14*	Improved programmable generators description [resolves #182].
58$^{(r3608)}$	*2018-03-01*	Updated dead links.
57$^{(r3206)}$	*2017-09-10*	Fixed API section invalid reference.
56$^{(r3172)}$	*2017-08-29*	Fixed Doxygen references.
55$^{(r3166)}$	*2017-08-29*	Updated Doxygen source files names.
54$^{(r3128)}$	*2017-08-13*	Adopted to: - Python 3.6; - Doxygen 1.8.13.
53$^{(r3000)}$	*2017-03-25*	Updated installation directory content description.
52$^{(r2772)}$	*2017-01-04*	Updated installation directory description for MATES 2.4.0.0.
51$^{(r2726)}$	*2016-12-27*	Added MATES-UCC-MK1 part.
50$^{(r2702)}$	*2016-12-22*	Corrected input statements for API.
49$^{(r2701)}$	*2016-12-22*	Added missing Python classes documentation.
48$^{(r2623)}$	*2016-10-12*	Updated directory structure section.
47$^{(r2392)}$	*2016-07-27*	Improved error reporting, documentation, further UCC device implementation.
46$^{(r2340)}$	*2016-06-02*	Removed the draft UCC part.
45$^{(r2339)}$	*2016-06-02*	Updated API reference.
44$^{(r2322)}$	*2016-05-19*	Added support for ADC readout for MATES-UCC-MK1.
43$^{(r2304)}$	*2016-04-26*	Added UCC part.
42$^{(r2201)}$	*2016-02-22*	Updated the CAN node address setting description [resolves #140].
41$^{(r2196)}$	*2016-02-21*	Updated FLASH organization description [resolves #141].
40$^{(r2143)}$	*2016-02-15*	Updated for Doxygen 1.8.11.
39$^{(r1789)}$	*2015-09-14*	Use SVN external to pull logo.
38$^{(r1761)}$	*2015-09-04*	Added DioAsserts interface.
37$^{(r1697)}$	*2015-06-22*	Updated installation directory description [resolves #119].
36$^{(r1678)}$	*2015-06-05*	Removed API documentation for abstract DioNode.
35$^{(r1677)}$	*2015-06-05*	Added another two modules to the API documentation.
34$^{(r1658)}$	*2015-05-26*	Added missing Dio node API reference.
33$^{(r1580)}$	*2015-03-28*	Added Dio3Node.cs API description.
32$^{(r1407)}$	*2014-12-22*	Updated installation directory description [resolves #83].
31$^{(r1339)}$	*2014-11-08*	Improved API documentation.
30$^{(r1325)}$	*2014-11-07*	Successfully merged with Doxygen documentation.
29$^{(r1240)}$	*2014-05-31*	Updated installation directory description.
28$^{(r1238)}$	*2014-05-31*	Completed the VISA section [resolves #77].
27$^{(r1151)}$	*2014-05-04*	Added MATES-DIO3-MK1 signal generators description.
26$^{(r1111)}$	*2014-04-28*	Added missing HEARTBEAT descriptions.
25$^{(r1109)}$	*2014-04-23*	Minor formatting corrections, added generic parameters tables for MATES-DAC5-MK1 and MATES-DIOX-MK1.
24$^{(r1106)}$	*2014-04-22*	Added MATES proxy diagram.
23$^{(r1103)}$	*2014-04-15*	Filled in the MATES-ADC5-MK1 accuracy value.
22$^{(r1100)}$	*2014-04-10*	Added installation directory listing.
21$^{(r1062)}$	*2014-04-03*	Completed ADC calibration section.
20$^{(r911)}$	*2014-01-30*	Added MK1 nodes memory map.
19$^{(r779)}$	*2013-12-21*	Major refactoring: - re-implemented segmented transfers to use only one monitor packet (support for atomic API operations for the MATES proxy server) - corrected synchronization in mates_loader_cmd to avoid null object access after a node is removed from the list.
18$^{(r758)}$	*2013-12-15*	Added description of the VX setting for MATES-DIOX-MK1.

Version	Date	Change descirpion
17$^{(r738)}$	*2013-12-08*	Added bootloader algorithm description.
16$^{(r704)}$	*2013-11-28*	Delivery documents development.
15$^{(r682)}$	*2013-11-26*	Added hardware overview section.
14$^{(r681)}$	*2013-11-26*	Work on Overview section.
13$^{(r628)}$	*2013-11-13*	Implemented setting outputs to default values using CR.ODEF bit.
12$^{(r598)}$	*2013-11-08*	Implemented reset request bit and DAC read operation.
11$^{(r563)}$	*2013-10-31*	Implemented DAC5 default outputs setting.
10$^{(r532)}$	*2013-10-22*	Minor improvements.
9$^{(r524)}$	*2013-10-18*	Removed some redundant content.
8$^{(r515)}$	*2013-10-16*	Initial implementation of the register access and MATES loader.
7$^{(r508)}$	*2013-10-08*	Corrected makefile.
6$^{(r491)}$	*2013-10-05*	Initially added register descriptions.
5$^{(r443)}$	*2013-09-13*	Added common datagrams.
4$^{(r356)}$	*2013-08-28*	Added HEARTBEAT datagram information.
3$^{(r341)}$	*2013-08-23*	Added some MATES-DIO3-MK1 parameters.
2$^{(r292)}$	*2013-08-05*	Added some general information.
1$^{(r277)}$	*2013-07-31*	Initial revision.

H Bibliography

References

[1] AT90CAN128.pdf, *8-bit Microcontroller with 32K/64K/128K Bytes of ISP Flash and CAN Controller*, Atmel, rev. 7679H–CAN–08/08. 384

[2] uOS_monitor_EN.pdf *uOS_monitor User's manual*, Viresco, rev. 1.9.

[3] MATES API, *MATES API Reference manual*, Viresco, rev. 2.5. 30, 36

[4] scpi-99.pdf, *Standard Commands for Programmable Instruments (SCPI)*, SCPI Consortium, VERSION 1999.0 May, 1999. 30

[5] IVI-3.1_Architecture_2014-03-28.pdf, *IVI-3.1: Driver Architecture Specification*, IVI Foundation, March 28, 2014 Edition Revision 3.5. 30

List of Tables

List of Figures

Listings