# RDS Interface

The Radio Data System transmits supplementary information in binary format, for example the station name or travel information, on an inaudible audio subcarrier of a radio program. This interface is aimed at devices capable of receiving and/or transmitting RDS information.

For more information see the core RDS standard :ref:`iec62106` and the RBDS standard :ref:`nrsc4`.

> **Note**
>
> Note that the RBDS standard as is used in the USA is almost identical to the RDS standard. Any RDS decoder/encoder can also handle RBDS. Only some of the fields have slightly different meanings. See the RBDS standard for more information.

The RBDS standard also specifies support for MMBS (Modified Mobile Search). This is a proprietary format which seems to be discontinued. The RDS interface does not support this format. Should support for MMBS (or the so-called 'E blocks' in general) be needed, then please contact the linux-media mailing list: https://linuxtv.org/lists.php.

## Querying Capabilities

Devices supporting the RDS capturing API set the `V4L2_CAP_RDS_CAPTURE` flag in the `capabilities` field of struct :c:type:`v4l2_capability` returned by the :ref:`VIDIOC_QUERYCAP` ioctl. Any tuner that supports RDS will set the `V4L2_TUNER_CAP_RDS` flag in the `capability` field of struct :c:type:`v4l2_tuner`. If the driver only passes RDS blocks without interpreting the data the `V4L2_TUNER_CAP_RDS_BLOCK_IO` flag has to be set, see :ref:`Reading RDS data <reading-rds-data>`. For future use the flag `V4L2_TUNER_CAP_RDS_CONTROLS` has also been defined. However, a driver for a radio tuner with this capability does not yet exist, so if you are planning to write such a driver you should discuss this on the linux-media mailing list: https://linuxtv.org/lists.php.

Unknown interpreted text role "c:type".

Whether an RDS signal is present can be detected by looking at the `rxsubchans` field of struct :c:type:`v4l2_tuner`: the `V4L2_TUNER_SUB_RDS` will be set if RDS data was detected.

Devices supporting the RDS output API set the `V4L2_CAP_RDS_OUTPUT` flag in the `capabilities` field of struct :c:type:`v4l2_capability` returned by the :ref:`VIDIOC_QUERYCAP` ioctl. Any modulator that supports RDS will set the `V4L2_TUNER_CAP_RDS` flag in the `capability` field of struct :c:type:`v4l2_modulator`. In order to enable the RDS transmission one must set the `V4L2_TUNER_SUB_RDS` bit in the `txsubchans` field of struct :c:type:`v4l2_modulator`. If the driver only passes RDS blocks without interpreting the data the `V4L2_TUNER_CAP_RDS_BLOCK_IO` flag has to be set. If the tuner is capable of handling RDS entities like program identification codes and radio text, the flag `V4L2_TUNER_CAP_RDS_CONTROLS` should be set, see :ref:`Writing RDS data <writing-rds-data>` and :ref:`FM Transmitter Control Reference <fm-tx-controls>`.

## Reading RDS data

RDS data can be read from the radio device with the :c:func:`read()` function. The data is packed in groups of three bytes.

## Writing RDS data

RDS data can be written to the radio device with the :c:func:`write()` function. The data is packed in groups of three bytes, as follows:

## RDS datastructures

```
.. c:type:: v4l2_rds_data
```

```
.. flat-table:: struct v4l2_rds_data
    :header-rows:  0
    :stub-columns: 0
    :widths:       1 1 5

    * - __u8
      - ``lsb``
      - Least Significant Byte of RDS Block
    * - __u8
      - ``msb``
      - Most Significant Byte of RDS Block
    * - __u8
      - ``block``
      - Block description
```

```
.. tabularcolumns:: |p{2.9cm}|p{14.6cm}|
```

```
.. flat-table:: Block description
    :header-rows:  0
    :stub-columns: 0
    :widths:       1 5
```

```
            * - Bits 0-2
              - Block (aka offset) of the received data.
            * - Bits 3-5
              - Deprecated. Currently identical to bits 0-2. Do not use these
                bits.
            * - Bit 6
              - Corrected bit. Indicates that an error was corrected for this data
                block.
            * - Bit 7
              - Error bit. Indicates that an uncorrectable error occurred during
                reception of this block.
```

```
    .. tabularcolumns:: |p{6.4cm}|p{2.0cm}|p{1.2cm}|p{7.0cm}|
```

```
    .. flat-table:: Block defines
        :header-rows:  0
        :stub-columns: 0
        :widths:       1 1 1 5

        * - V4L2_RDS_BLOCK_MSK
          -
          - 7
          - Mask for bits 0-2 to get the block ID.
        * - V4L2_RDS_BLOCK_A
          -
          - 0
          - Block A.
        * - V4L2_RDS_BLOCK_B
          -
          - 1
          - Block B.
        * - V4L2_RDS_BLOCK_C
          -
          - 2
          - Block C.
        * - V4L2_RDS_BLOCK_D
          -
          - 3
          - Block D.
        * - V4L2_RDS_BLOCK_C_ALT
          -
          - 4
          - Block C'.
        * - V4L2_RDS_BLOCK_INVALID
          - read-only
          - 7
          - An invalid block.
        * - V4L2_RDS_BLOCK_CORRECTED
          - read-only
          - 0x40
          - A bit error was detected but corrected.
        * - V4L2_RDS_BLOCK_ERROR
          - read-only
          - 0x80
          - An uncorrectable error occurred.
```