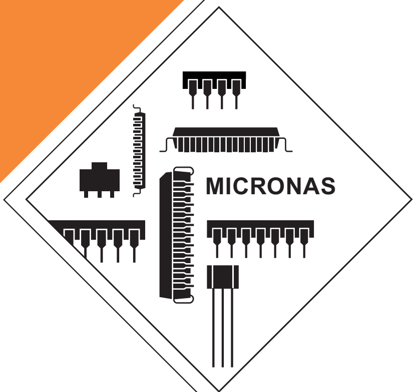


DATA SHEET

DRX 8872C
COFDM Demodulator/FEC

Edition Feb. 18, 2004
6251-618-2DS

MICRONAS

Contents

Contents, continued

COFDM Demodulator/FEC

1. Introduction

The DRX 8872C is an ETS 300 744-compliant integrated demodulator and forward error corrector (FEC) for DVB-T receivers.

The IC accepts 1st and 2nd IF COFDM signals as input data. The 1st IF sampling option further decreases system cost. The incoming signal is sampled by a high-performance 10-bit A/D converter. The internal microprocessor performs the detection of the COFDM parameters and configuration of the demodulator automatically, without any interaction with the host processor. The error correction unit corrects remaining errors and outputs a DVB-compliant MPEG-2 transport stream.

The DRX 8872C can cope with very severe channel distortions due to its state-of-the-art channel estimation unit.

1.1. Features

– Excellent performance in presence of echoes, co-channel and AWGN
– Integrated microprocessor to perform autonomous operation
– Flexible concept by “micro-coded” algorithms
– Detection of channel type (echoes, co-channel, gaussian noise …) via channel classificator function
– Very suitable for SFN operation
– Complete software API for smooth integration
– Integrated 10-bit ADC
– 1st and 2nd IF COFDM supported
– No VCXO required because of digital resampling techniques
– Quick synchronization after channel switch (<70 ms)
– 6 MHz, 7 MHz, and 8 MHz channel-compliant with only one crystal
– Supports all DVB-T modes including hierarchical modulation
– Digital AFC
– BER, S/N, packet error, constellation diagram, lock indication readout
– Serial or parallel MPEG-2 transport stream output
– Supply voltage: 2.5 V (core); 3.3V (I/O)
– Control: via serial bus
– Package: 80-pin PTQFP
– Ambient operating temperature: 0 °C to +70 °C
– IEEE 1149.1 boundary scan

1.2. Applications

– IDTV receivers
– Set-top boxes
– Network Interface Modules (NIMs)
– PC-TV cards

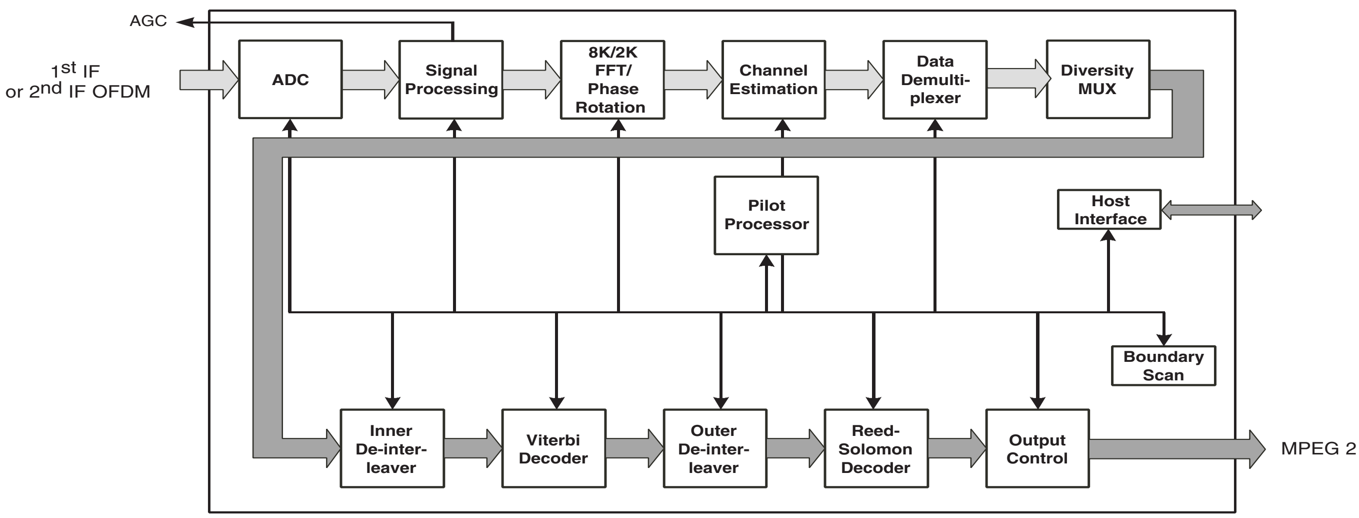


Fig. 1–1: Block diagram of the DRX 8872C

1.3. DVB-T Front-End Application

Fig. 1–2 shows a block diagram of a typical DVB-T front-end using the DRX 8872C at the first intermediate frequency. The tuner converts the COFDM signal to a first intermediate frequency of about 36 MHz which is then band-pass-filtered by a SAW filter stage. The SAW filter is followed by a differential-in / differential-out IF amplifier.

This chapter describes in detail how to connect the DRX 8872C in such an application.

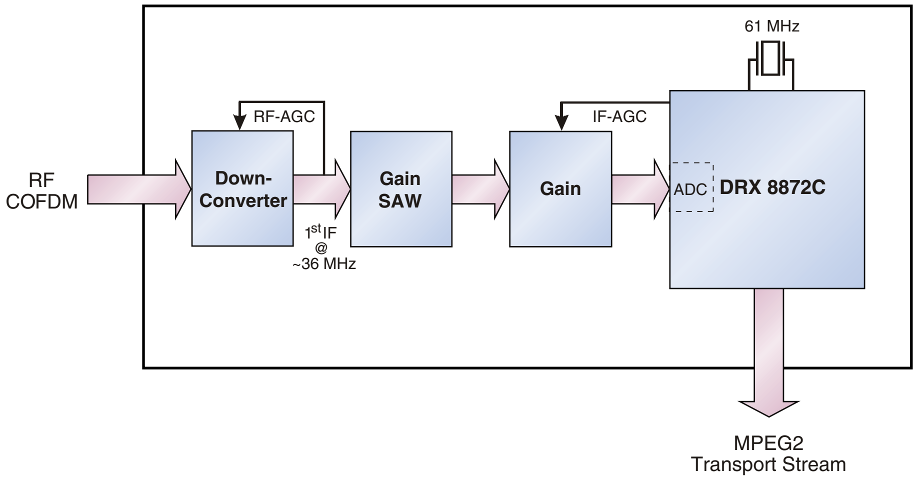


Fig. 1–2: Front-end block diagram for the DRX 8872C

2. Functional Description

2.1. Analog input

The input signal of the DRX 8872C consists of a 1st or 2nd IF COFDM signal with a bandwidth of 6, 7, or 8 MHz. The input of the demodulator is differential for optimal noise performance. The reference voltages that are needed for the ADC are generated internally. The reference voltages are connected to pins for decoupling reasons.

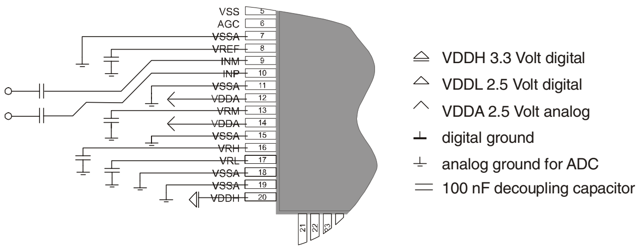


Fig. 2–1: Input stage application diagram.

2.2. AGC

The DRX 8872C delivers a control signal for an amplifier in the tuner part of the system. The system controller controls a register that is used for a pulse train using a Pulse Width Modulator (PWM) circuit. 256 values can be distinguished. The SP7_SetAGCParams function of the API determines the way this register is controlled. By default, the maximum level is represented by all ‘1’s and the minimum level is represented by all ‘0’s. A low-pass RC filter connected to the AGC pin will filter the PWM signal, generating a stable analog signal for controlling the amplifier of the tuner. The bandwidth of this filter should be small enough to minimize the PWM-jitter on the control signal. The external AGC control is slow compared to the fast internal AGC. A filter bandwidth of 1 kHz is recommended for filtering the PWM-jitter.

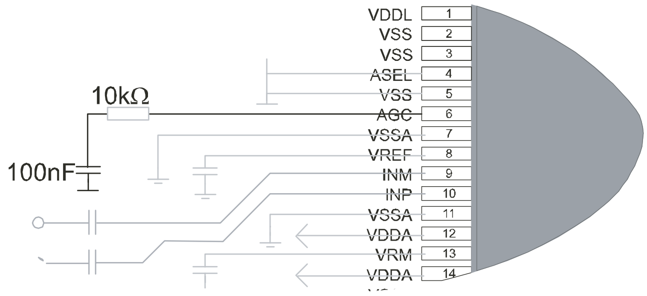


Fig. 2–2: AGC application diagram.

For tuners that need voltages other than 0 to 3.3 Volt a buffer connected to 5 Volt could be inserted for generating a higher control voltage.

2.3. Crystal

A single crystal operates the DRX 8872C. Sample rate mismatches are corrected completely in the digital domain. The clock generated by the crystal is used as the system clock. A ‘divide by three’-block generates the sample clock for the ADC. There are two possible frequencies that can be used by the DRX 8872C. When using the 1st IF sampling technique, a 61 MHz crystal should be used. For 2nd IF sampling one should select a 55 MHz crystal. The next figure shows why.

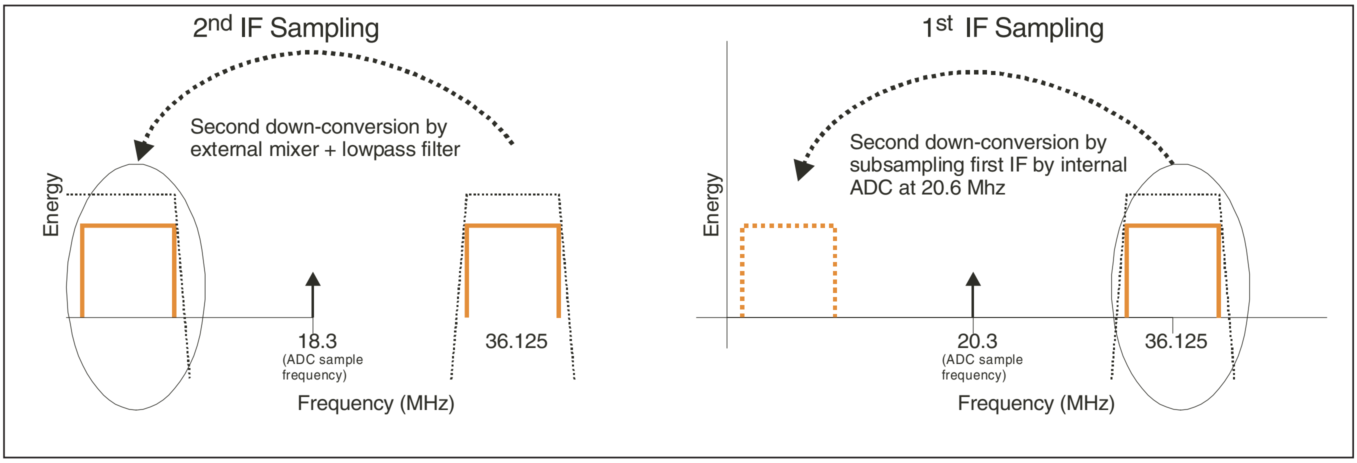


Fig. 2–3: 1st or 2nd IF sampling

The API function SP7_SetSamplingMode has been defined to set the mode. Default 1st IF sampling is assumed. To generate 55 or 61 MHz, a third overtone crystal must be used. The fundamental tone must be filtered out using an LC network with a time constant defined by:

$$\omega = \frac{1}{\sqrt{LC}}, \text{ where } \omega = \frac{1}{2\pi f}$$

with f=61 MHz for 1st IF sampling and f=55 MHz for baseband sampling. The application diagram shows the complete crystal circuitry, for example:

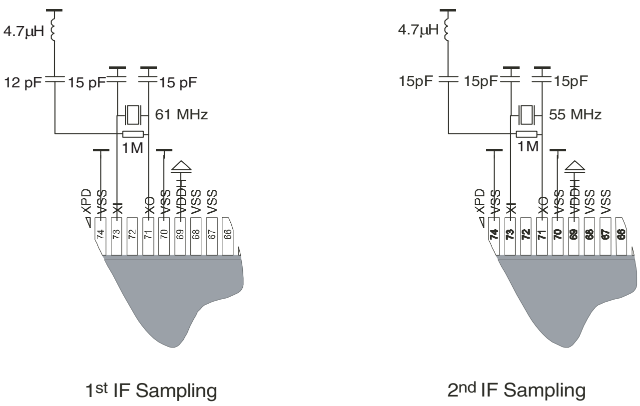


Fig. 2–4: Clocking circuitry for 1st IF or baseband sampling

A frequency mismatch of ±100 ppm is allowed for the described crystals.

2.4. Manufacturing Guidelines

To maximize both the removal of heat from the package and the electrical performance, a land pattern must be incorporated on the PCB within the footprint of the package corresponding to the exposed metal pad on the package, as shown in Fig. 2–5.

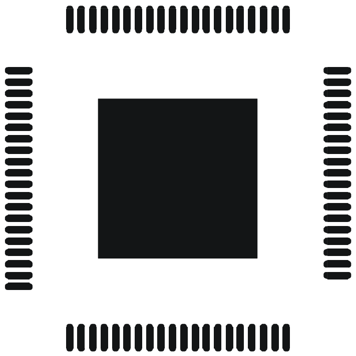


Fig. 2–5: Top layer solder mask

The dimensions of the PCB pad may be larger or smaller or even a different shape than the exposed metal pad but should have a clearance of at least 0.25 mm between the outer edges of the land pattern and the inner edges of pad pattern for the leads to avoid any shorts.

To improve the thermal dissipation, a thermal via array should be made within the PCB pad area. The vias should be 0.3 mm in diameter at a pitch of between 1.0 and 1.2 mm and preferably with 1 oz copper via barrel plating.

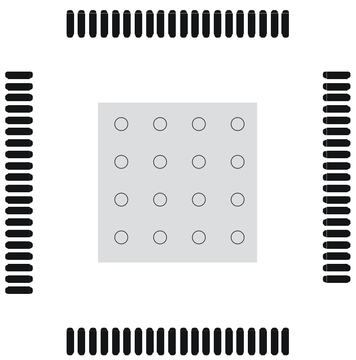


Fig. 2–6: Thermal via array

For further information regarding the optimal usage of the Micronas exposed-pad QFP please refer to the Application Note “Surface Mount Assembly of Exposed-Pad QFP packages”.

2.5. Lock Indication

Two hardware lock indicators are available. They can be used for LED control as shown in Fig. 2–7.

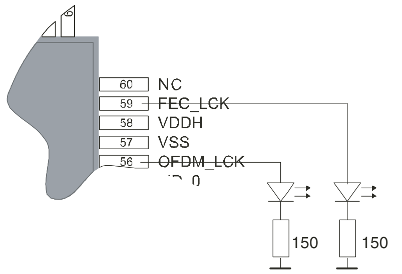


Fig. 2–7: Lock indication LEDs

The lock indicator pins FEC_LCK and OFDM_LCK can also be put in two other modes:

SAW filter select

In this mode the two lock signal indicate whether the input signal has a 6, 7, or 8 MHz bandwidth. These pins can than be used directly to switch SAW filters in the tuner accordingly.

User defined

In this mode the value of the pins can be programmed via software by the host of the system.

The API function SP7_LCKMode takes care of these settings.

2.5.1. Host Interface

The DRX 8872C communicates via a serial protocol. The DRX 8872C only acts as a slave device. A write access consists of a 'start' followed by the DRX 8872C device address, then the internal register address (2 bytes) and finally the data that needs to be written. During a read-access a repeated start should follow the internal register address definition, after which the data can be read.

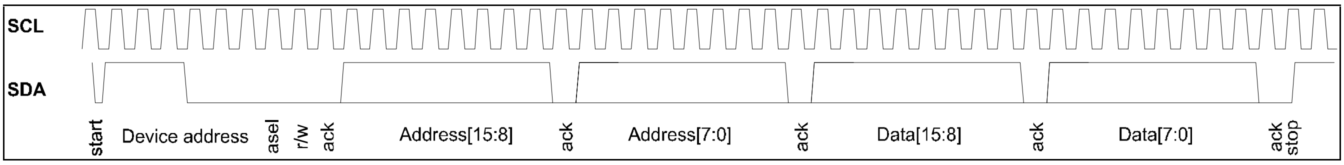


Fig. 2–8: I2C_Write timing (NrOfBytes = 2)

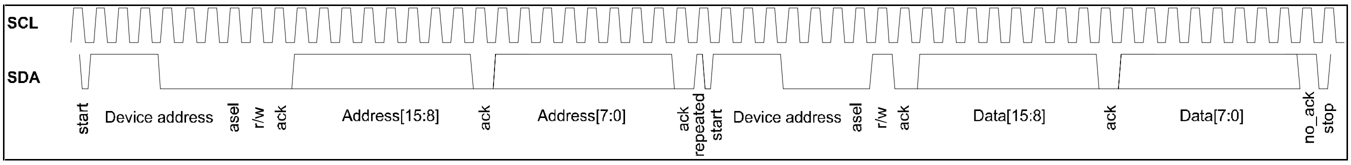


Fig. 2–9: I2C_Read timing (NrOfBytes = 2)

The device address is either "1110000x" if the ASEL pin is tied low or "1110001x" if the ASEL pin is tied high.

2.6. MPEG2 Output

The DRX 8872C has a parallel or serial transport stream output. The system controller adjusts the MPEG2 clock (MCLK) in order to minimize the jitter on the gaps in between packets. The maximum jitter is limited to 20 µs. There are several combinations of clock and data formats possible allowing flexible interfacing to MPEG2 decoders. The following figure shows the behavior of the MPEG2 output pins in parallel and serial mode.

The DRX 8872C can output the parity bytes in between two transport stream packages or one can choose to have the transport stream packets output without parity bytes. The API function SP7_TSOutput lets the user control this. Furthermore, this function can be used to put the MPEG output pins in tristate mode. This can be useful in multistandard STBs where both satellite and terrestrial front-ends are connected to one common interface. No external multiplexer is needed.

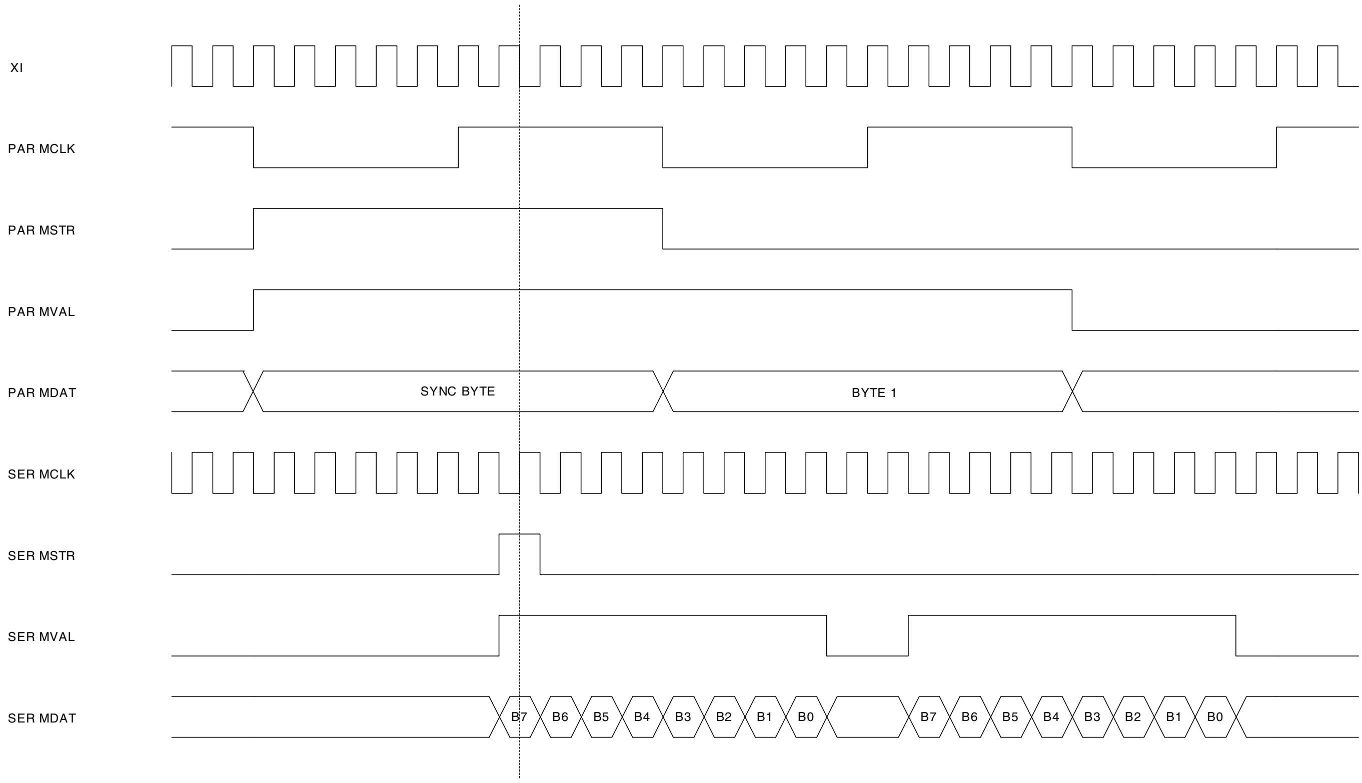


Fig. 2–10: MPEG2 TS output timing

3. Application Programming Interface (API)

For a description of the types used in the API prototypes, refer to Appendix A in Section 5. on page 37. Most functions in the API return a status (of type Status_t) that indicates whether the function completed successfully.

The API assumes availability of low-level serial interface functions, as described in Appendix B.

3.1. Initialize System Settings

The DRX 8872C integrates a processor. This processor performs the algorithms to synchronize and track the COFDM signal. The algorithms are described in microcode.

There are two versions of the microcode available: A basic version including only the code that will actually be uploaded to the chip and another version including a so-called symbol-table. The basic version is 16 kB and the extended code is about 20 kB. The symbol table includes a memory map of internal processor variables. When monitoring these variables, the symbol table tells the application which address to use. Most applications don’t need to monitor the internal processor variables and then one only needs the basic code. In the API distribution, the source code for a tool called “SP7Strip” is included. This tool can strip the symbol table from the microcode as distributed.

After start of the software application, uploading the microcode to the internal processors memory must be the first action. When uploading the originally distributed microcode, one should use the following function.

3.1.1. SP7_LoadMicrocode

```
Status_t    SP7_LoadMicrocode   (     pu16_t      mc_addr,
                                      pu32_t      version
                                );
```

Variable	Type	Description
mc_addr	pu16_t	Memory address of the binary file containing the algorithms for the internal processor.
version	pu32_t	Returns version of microcode uploaded.

For uploading the ‘stripped’ version, the following function is available.

3.1.2. SP7_LoadImage

```
Status_t    SP7_LoadImage       (     u16_t       image_size,
                                      pu8_t       image_addr,
                                      pu32_t      version
                                );
```

Variable	Type	Description
image_size	u16_t	Number of bytes to load (typically 16384).
image_addr	pu8_t	Memory address of the 16 kB stripped binary file containing the algorithms for the internal processor.
version	pu32_t	Returns version of microcode uploaded.

The processor inside the DRX 8872C needs to be given some basic information about the application in order to be able to automatically acquire lock. The first parameter is the sampling mode. The DRX 8872C supports 1st IF sampling as well as base-band sampling. All one needs to do is tell the processor which system clock frequency (crystal) and which input signal frequency – generally 36.125 MHz for 1st IF or 4.57 MHz for baseband sampling – is used. In combination with knowledge about whether the signal is mirrored or not, the chip will automatically calculate the correct settings for the internal sample-rate correction filters. The digital timing recovery enables the system to acquire lock when the system clock is within 500 ppm of the preprogrammed value.

3.1.3. SP7_SetSamplingMode

```
Status_t    SP7_SetSamplingMode (     u16_t       SystemClockFrequency,
                                      u16_t       SigInputFrequency,
                                      Mirror_t    SigInputMirror
                                 );
```

Variable	Type	Description
SystemClockFrequency	u16_t	Frequency of the crystal applied to the DRX 8872C, in kHz (e.g. set to 61000 when using a 61-MHz crystal).
SigInputFrequency	u16_t	16 bit integer number representing the center frequency of the COFDM signal that is applied to the DRX 8872C, in kHz (e.g. set to 36125 for 1st IF sampling).
SigInputMirror	Mirror_t	In case the input signal to the DRX 8872C is mirrored, depending on the number of down-conversion stages in the system, this can be indicated using this flag. The DRX 8872C will adjust automatic frequency correction algorithms accordingly.

Depending on the tuner one uses, it is either preferred to have a slow AGC control or a fast feedback to changes on the incoming energy level. The type of control can be selected using the AGC control variable in the next API function.

When not controlled by the internal processor, it is possible to adjust the level of the amplifier stages by changing the AGC-level variables as described in the next function.

3.1.4. SP7_SetAGCParams

```
Status_t    SP7_SetAGCParams    (    AGC_t       AGC_ctrl,
                                     Bool_t      AGC_SelectPin2,
                                     u16_t       AGC_level1,
                                     u16_t       AGC_level2,
                                     u16_t       AGC_level3
                                );
```

Variable	Type	Description
AGC_ctrl	AGC_t	Variable indicating type of control. AGC_OFF No control, the default levels will be applied. AGC_FAST Fast reaction to changes in energy level. In this mode the chip uses a hardwired AGC control function. AGC_SLOW Instead of the hardwired AGC functionality the internal processor controls the incoming energy level on symbol basis. AGC_INV By default, a positive control is used. With this setting this can be changed into a negative control. This is only possible in combination with AGC_SLOW. If one wants to negate the control in combination with AGC_FAST, one must use an external inverting component.
AGC_SelectPin2	Bool_t	If TRUE, AGC is applied to the secondary PWM output (only DRX8872P).
AGC_level1	u16_t	Unsigned integer indicating the default level of AGC1 when no control is applied to it.
AGC_level2	u16_t	Unsigned integer indicating the default level of AGC2 when no control is applied to it.
AGC_level3	u16_t	Unsigned integer indicating the default level of AGC3 when no control is applied to it.

Upon power-on reset, all output pins are tri-stated to allow other demodulators (e.g. satellite or cable) to drive the TS signals. The output pins must be put in output mode using the following API function. The monitor-bus with real-time channel information is only available with the professional version.

3.1.5. SP7_SetOutputEnable

```
Status_t    SP7_SetOutputEnable (     Bool_t      MpegTS,
                                      Bool_t      MonitorBus,
                                      Bool_t      InterfacePins
                                );
```

Variable	Type	Description
MpegTS	Bool_t	Boolean indicating MPEG TS pin driver state. FALSE MPEG TS pins are tristated (default). TRUE MPEG TS pins are put in output mode.
MonitorBus	Bool_t	Boolean indicating Monitor bus pin driver state (MT8872P only). FALSE Monitor bus pins are tristated (default). TRUE Monitor bus pins are put in output mode.
InterfacePins	Bool_t	Boolean indicating pin driver state of OFDM_LCK, FEC_LCK, I2C2_EN, IRQN and SCL2 (DRX 8872C only). FALSE Pins are tristated. TRUE Pins are put in output mode.

The actual mode of the pins can be obtained using the following function.

3.1.6. SP7_GetOutputEnable

```
Status_t    SP7_GetOutputEnable (     pBool_t     MpegTS,
                                      pBool_t     MonitorBus,
                                      pBool_t     InterfacePins
                                );
```

Variable	Type	Description
MpegTS	pBool_t	Boolean indicating MPEG TS pin driver state. FALSE MPEG TS pins are tristated (default). TRUE MPEG TS pins are in output mode.
MonitorBus	pBool_t	Boolean indicating Monitor bus pin driver state (MT8872P only). FALSE Monitor bus pins are tristated (default). TRUE Monitor bus pins are in output mode.
InterfacePins	pBool_t	Boolean indicating pin driver state of OFDM_LCK, FEC_LCK, I2C2_EN, IRQN and SCL2 (DRX 8872C only). FALSE Pins are tristated. TRUE Pins are in output mode.

The MPEG Transport Stream interface clocking can be configured. When no valid data bytes are output (the parity bytes in between two packets or during the remaining gap in the guard interval) the MCLK pin can either be disabled or a continuous clock can be chosen. When using the continuous clock the MVAL signal can be used to indicate non-valid data.

3.1.7. SP7_TSOutput

```
Status_t    SP7_TSOutput        (    Bool_t    Parity,
                                     Bool_t    SuppressClock
                                );
```

Variable	Type	Description
Parity	Bool_t	If TRUE, The Reed Solomon parity bytes will be output in between two transport stream packets.
SuppressClock	Bool_t	If TRUE, the MCLK pin is suppressed when no valid data is output (parity bytes and gaps between packets or during guard interval).

The last pins that can be configured are the OFDM_LCK and the FEC_LCK pins.

3.1.8. SP7_LCKMode

```
Status_t    SP7_LCKMode         (    LCK_t     OFDM_LCK_Ctrl,
                                     LCK_t     FEC_LCK_Ctrl
                                );
```

Variable	Type	Description
OFDM_LCK_Ctrl	LCK_t	Parameter indicating use of the OFDM LCK pin. LCK_INDICATOR The OFDM_LCK pin reflects the locking state of the COFDM demodulator part of the chip. LCK_SAW_8 The OFDM_LCK pin will go high if an 8 MHz SAW filter must be selected, otherwise (for 7 or 6 MHz bandwidths) it will be low. LCK_SAW_7 The OFDM_LCK pin will go high if an 7 or 8 MHz SAW filter must be selected, otherwise (for 6 MHz bandwidths) it will be low. LCK_UIO_0 The OFDM_LCK pin will be always low. LCK_UIO_1 The OFDM_LCK pin will be always high.
FEC_LCK_Ctrl	LCK_t	Parameter indicating use of the FEC LCK pin. LCK_INDICATOR The FEC_LCK pin reflects the locking state of the FEC error correction part of the chip. LCK_SAW_8 The FEC_LCK pin will be high if an 8 MHz SAW filter must be selected, otherwise (for 7 or 6 MHz bandwidths) it will be low. LCK_SAW_7 The FEC_LCK pin will go high if an 7 or 8 MHz SAW filter must be selected, otherwise (for 6 MHz bandwidths) it will be low. LCK_UIO_0 The FEC_LCK pin will be always low. LCK_UIO_1 The FEC_LCK pin will be always high.

For function *SP7_LCKMode* to work correctly, the OFDM_LCK and FEC_LCK pins must have been configured for output, by using the function *SP7_SetOutputEnable.*

3.2. Setup Signal Parameters

All hardware related parameters have been set now. The next step is to configure the system for the incoming signal. In the application one can either let the DRX 8872C detect all settings automatically or one can choose to pre-program some parameters in case they are known. This can improve locking speed. The functions *SP7_GetChannelParams* and *SP7_SetChannelParams* relate to channel parameters like frequency offsets and signal bandwidth. The *SP7_GetOfdmParams* and *SP7_SetOfdmParams* functions relate to the COFDM symbol parameters like code rate, FFT size etc. The parameters that are set up with these functions are not programmed until the chip is started using the *SP7_Start* function.

3.2.1. SP7_SetChannelParams

```
Status_t    SP7_SetChannelParams(     s16_t       SysFrequencyOffset,
                                      s16_t       SigFrequencyOffset,
                                      Bandwidth_t SigBandwidth,
                                      Mirror_t    SigMirror,
                                      Cls_t       SigClass
                                 );
```

Variable	Type	Description
SysFrequencyOffset	s16_t	Signed integer number indicating the known system clock frequency mismatch in kHz, relative to the value that was set with *SP7_SetSamplingMode*.
SigFrequencyOffset	s16_t	Signed integer number indicating the known signal frequency mismatch in kHz. If unknown, set to zero.
SigBandwidth	Bandwidth_t	The bandwidth can be programmed. BANDWIDTH_6MHZ 6 MHz Channel. BANDWIDTH_7MHZ 7 MHz Channel. BANDWIDTH_8MHZ 8 MHz Channel (default).
SigMirror	Mirror_t	This parameter indicates whether the spectrum is mirrored. MIRRORED Spectrum is mirrored. NORMAL Spectrum is normal. AUTO Detect automatically.
SigClass	Cls_t	This parameter sets the channel classification. CLS_GAUSS Gaussian noise. CLS_HEAVYGAUSS Heavy Gaussian noise. CLS_COCHANNEL Co-channel. CLS_STATIC Static echo. CLS_MOVING Moving echo. CLS_ZERODB Zero dB echo. AUTO Detect automatically.

Next to programming the values, the measured values can be obtained from the system.

3.2.2. SP7_GetChannelParams

```
Status_t     SP7_GetChannelParams(     ps16_t      SysFrequencyOffset,
                                        ps16_t      SigFrequencyOffset,
                                        pBandwidth_tSigBandwidth,
                                        pMirror_t   SigMirror,
                                        pCls_t      SigClass
                                      );
```

Variable	Type	Description
SysFrequencyOffset	ps16_t	Signed integer number indicating the system clock frequency mismatch in kHz, relative to the value that was set with *SP7_SetSamplingMode*.
FrequencyOffset	ps16_t	Signed integer number indicating the signal frequency mismatch in kHz. Depending on the step-size of the tuner, the mismatch can be minimized by re-tuning.
SigBandwidth	pBandwidth_t	The bandwidth is reported. This returns either a default value or the value that was set by a previous call of SP7_SetChannelParams. BANDWIDTH_6MHZ 6 MHz Channel. BANDWIDTH_7MHZ 7 MHz Channel. BANDWIDTH_8MHZ 8 MHz Channel.
SigMirror	pMirror_t	This parameter indicates whether the spectrum is mirrored. MIRRORED Spectrum is mirrored. NORMAL Spectrum is normal.
SigClass	pCls_t	The channel classification is reported. CLS_GAUSS Gaussian noise. CLS_HEAVYGAUSS Heavy Gaussian noise. CLS_COCHANNEL Co-channel. CLS_STATIC Static echo. CLS_MOVING Moving echo. CLS_ZERODB Zero dB echo.

The mode of the COFDM signal path is being transmitted using the TPS carriers can be detected automatically. The information can be obtained by using the *SP7_GetOfdmParams* function and the system can be told to use a specific mode using the *SP7_SetOfdmParams* API function.

3.2.3. SP7_SetOfdmParams

```
Status_t    SP7_SetOfdmParams    (    Mode_t     Mode,
                                      Guard_t    Guard,
                                      Const_t    Constellation,
                                      Hier_t     Hierarchy,
                                      Prior_t    Priority,
                                      Rate_t     CodeRate
                                 );
```

Variable	Type	Description
Mode	Mode_t	MODE_2K 8k Mode. MODE_8K 2k Mode. AUTO Detect automatically.
Guard	Guard_t	GUARD_32 1/32nd Guard interval. GUARD_16 1/16th Guard interval. GUARD_8 1/8th Guard interval. GUARD_4 1/4th Guard interval. AUTO Detect automatically.
Constellation	Const_t	CONST_QPSK QPSK constellation. CONST_QAM16 QAM16 constellation. CONST_QAM64 QAM64 constellation. AUTO Detect automatically.
Hierarchy	Hier_t	HIER_NONHIER No hierarchical transmission. HIER_ALPHA_1 Hierarchical transmission, α is one. HIER_ALPHA_2 Hierarchical transmission, α is two. HIER_ALPHA_4 Hierarchical transmission, α is four. AUTO Detect automatically.

Variable	Type	Description
Priority	Prior_t	In case of hierarchical transmission: PRIOR_LOW Low priority. PRIOR_HIGH High priority.
CodeRate	Rate_t	RATE_1_2 Code rate 1/2nd. RATE_2_3 Code rate 2/3rd. RATE_3_4 Code rate 3/4th. RATE_5_6 Code rate 5/6th. RATE_7_8 Code rate 7/8th. AUTO Detect automatically.

3.2.4. SP7_GetOfdmParams

```
Status_t    SP7_GetOfdmParams    (     pMode_t     Mode,
                                       pGuard_t    Guard,
                                       pConst_t    Constellation,
                                       pHier_t     Hierarchy,
                                       pPrior_t    Priority,
                                       pRate_t     CodeRate
                                 );
```

Variable	Type	Description
Mode	pMode_t	MODE_2K 8k Mode. MODE_8K 2k Mode.
Guard	pGuard_t	GUARD_32 1/32nd Guard interval. GUARD_16 1/16th Guard interval. GUARD_8 1/8th Guard interval. GUARD_4 1/4th Guard interval.
Constellation	pConst_t	CONST_QPSK QPSK constellation. CONST_QAM16 QAM16 constellation. CONST_QAM64 QAM64 constellation.
Hierarchy	pHier_t	HIER_NONHIER No hierarchical transmission. HIER_ALPHA_1 Hierarchical transmission, α is one. HIER_ALPHA_2 Hierarchical transmission, α is two. HIER_ALPHA_4 Hierarchical transmission, α is four.
Priority	pPrior_t	In case of hierarchical transmission: PRIOR_LOW Low priority. PRIOR_HIGH High priority.
CodeRate	pRate_t	RATE_1_2 Code rate 1/2nd. RATE_2_3 Code rate 2/3rd. RATE_3_4 Code rate 3/4th. RATE_5_6 Code rate 5/6th. RATE_7_8 Code rate 7/8th.

The processor now knows all it needs to know and one can start to acquire lock.

The *SP7_Start* function will start the internal processor and the demodulation process will start, making use of the settings that were applied using the previously described API functions.

3.2.5. SP7_Start

```
Status_t    SP7_Start           (     void );
```

If an COFDM signal is applied to the input of the DRX 8872C, the system will start to acquire lock.

3.3. Selecting Channels

In order to get a correct COFDM signal, the tuner in front of the demodulator must be programmed to the right frequency. The DRX 8872C has been tested in combination with many tuners. Each tuner has its own specifications. The frequency range, step-size and charge pump settings vary per tuner. Also the optimal AGC setting can be different per tuner type. To ease the work of the customers some API functions have been made available. When using one of the tested tuner types the optimal settings will automatically be chosen.

To initialize the API with the settings of the tuner that is currently in use, first the *SP7_Tune_init* function should be called.

3.3.1. SP7_Tune_init

```
Status_t    SP7_Tune_init       (     u16_t       tune_type );
```

Variable	Type	Description
tune_type	u16_t	Variable indicating which tuner is used. Refer to source code for list of tuners supported.

The function *SP7_Tune_properties* can be called to retrieve the parameters of the tuner that is used (if known by the API). The output of this function can be used when setting up the system. In this case the user does not have to worry about whether the tuner outputs its signal at 1st or 2nd IF, whether the signal is mirrored or not and what the preferred AGC settings should be.

3.3.2. SP7_Tune_properties

```
Status_t    SP7_Tune_properties (    pu8_t*      Name,
                                     pu32_t      FrequencyMin,
                                     pu32_t      FrequencyMax,
                                     pu16_t      FrequencyStep,
                                     pu16_t      FrequencyOut,
                                     pMirror_t   FrequencyMirror,
                                     pAGC_t      AgcType,
                                     pu16_t      Agc_level3
                                );
```

Variable	Type	Description
Name	pu8_t*	Returns the name of the tuner. Only relevant when more than one tuner is taken into account during compilation, for instance in applications like Signal Spyder.
FrequencyMin	pu32_t	Minimum available frequency for this tuner, in kHz.
FrequencyMax	pu32_t	Maximum supported frequency for this tuner, in kHz.
FrequencyStep	pu16_t	RF frequency step-size for this tuner.
FrequencyOut	pu16_t	Output frequency of this tuner.
FrequencyMirror	pMirror_t	Variable indicating whether the output of this tuner is mirrored or not.
AgcType	pAGC_t	Variable indicating the preferred AGC setting for the tuner currently used.
AGC_level3	pu16_t	Integer value indicating the optimum PGA level in combination with the tuner currently used.

Next to acquiring information related to the type of tuner used, the API supports functions that will actually program the tuner to the desired frequency with the appropriate settings.

3.3.3. SP7_Tune_program

```
Status_t    SP7_Tune_program    (    u32_t      F );
```

Variable	Type	Description
F	u32_t	Frequency to which the tuner should be programmed, in kHz.

In case communication with the tuner fails, the exact error status can be read back using the following function.

3.3.4. SP7_Tune_error

```
u16_t       SP7_Tune_error      (    void );
```

Returns number that represents an error code.

Since the tuner has a finite step-size it will not be possible to exactly program the tuner to the desired frequency. The next function reports the frequency to which the tuner was actually programmed.

3.3.5. SP7_Tune_get_freq

```
u32_t        SP7_Tune_get_freq    (     void );
```

Returns exact frequency to which the tuner was programmed, in kHz.

When not using the given tuner programming functions, one needs to calculate the tuner settings and generate the command string (5 bytes) using the tuner specification. The DRX 8872C supports a gated clock-line to the tuner. This means that the tuner is not connected directly to the serial bus but via the demodulator and an external analog switch. If connected in this way, the serial lines towards the tuner can be kept quiet during normal operation to minimize noise on the PLL inside the tuner. To select a channel first the connection to the tuner must be enabled, then the tuner specific settings must be applied (5 I^2C byte accesses), and finally the connection to the tuner can be closed again. The following API function has been defined for opening and closing this port.

3.3.6. SP7_EnableTunerAccess

```
Status_t     SP7_EnableTunerAccess(     Bool_t      TunerAccess );
```

Variable	Type	Description
TunerAccess	Bool_t	If TRUE, the secondary protocol port is opened for programming the tuner.

After programming the tuner, the chip will start to acquire lock. It is of course important to know whether the chip has acquired lock. For this purpose the *SP7_LockingStatus* function has been defined.

3.3.7. SP7_LockingStatus

```
Status_t     SP7_LockingStatus    (     pBool_t     Locked );
```

Variable	Type	Description
Locked	pBool_t	If TRUE, chip has acquired lock.

3.4. Monitor Channel Quality

The channel estimator unit of the DRX 8872C performs a signal to noise measurement on the constellation diagram. The output of this calculation can be retrieved using the API function *SP7_GetSN*. From this measurement, a MER is also derived. The MER is calculated using the deviation of the continual pilots and is therefore an indication, not an exact value.

3.4.1. SP7_GetSN

```
Status_t    SP7_GetSN           (     pu16_t      SN,
                                      pu16_t      MER
                                );
```

Variable	Type	Description
SN	pu16_t	Integer number indicating the S/N ratio detected at the input of the demapper, in steps of 1/10th of a dB.
MER	pu16_t	Integer number indicating the MER, in steps of 1/10th of a dB.

The S/N ratio gives a very good indication of the channel quality, independent of the mode of the signal. For a graphical feedback one can use the constellation diagram. A slow non-real-time diagram can be built using the following function.

3.4.2. SP7_GetConstellationDiagram

```
Status_t    SP7_GetConstellationDiagram (
                                      ps16_t      Real,
                                      ps16_t      Imag
                                );
```

Variable	Type	Description
Real	ps16_t	Real part of the constellation point. The values 256/-256 correspond to the energy level of the pilots of the COFDM symbol. The maximum values range from –512 to +511.
Imag	ps16_t	Imaginary part of the constellation point.

A more common way to give an indication of the signal quality is to look at the BER values. This also allows for comparison of the quality of the DRX 8872C to the ETS300744 DVB-T standard.

3.4.3. SP7_GetBer

```
Status_t    GetBer              (     pu32_t      PreViterbi,
                                      pu32_t      PostViterbi,
                                      pu32_t      PacketError
                                );
```

Variable	Type	Description
PreViterbi	pu32_t	Bit error rate (BER) before error correction, with a scale of 1e-6.
PostViterbi	pu32_t	BER after Viterbi decoder, with a scale of 1e-6. A value of 200 corresponds to a BER of 2e-4, the so-called QEF level.
PacketError	pu16_t	Counter indicating the number of packet errors after Reed Solomon decoding.

Next to using the previously described function *SP7_GetOfdmParams* to obtain information on the incoming COFDM signal, the TPS information contained in the COFDM signal can be read using the following API function.

3.4.4. SP7_GetTpsInfo

```
Status_t    SP7_GetTpsInfo      (     pMode_t     TpsMode,
                                      pGuard_t    TpsGuard,
                                      pConst_t    TpsConstellation,
                                      pHier_t     TpsHierarchy,
                                      pRate_t     TpsHiRate,
                                      pRate_t     TpsLoRate,
                                      pTpsFrame_t TpsFrame,
                                      pu8_t       TpsLength,
                                      pBool_t     TpsCellIdRdy,
                                      pu16_t      TpsCellId
                                );
```

Variable	Type	Description
TPSMode	pMode_t	Variable indicating the mode as described in the TPS parameters.
TpsGuard	pGuard_t	Guard interval length as described in the TPS parameters.
TpsConstellation	pConst_t	Constellation diagram depth as described in the TPS parameters.
TpsHierarchy	pHier_t	Level of Hierarchy as described in the TPS parameters.
TpsHiRate	pRate_t	High priority code rate.
TpsLoRate	pRate_t	Low priority code rate.
TpsFrame	pTpsFrame_t	Frame number.
TpsLength	pu8_t	Integer number indicating the length of the TPS Frame.
TpsCellIdRdy	pBool_t	Flag indicating that the returned TpsCellId is valid.
TpsCellId	pu16_t	16-Bit received Cell identifier.

3.5. Commands and other Useful Functions

Next to the application specific function as described in the previous paragraphs, also some low level functions are available. Other API functions also make use of these low-level functions themselves.

3.5.1. SP7_Reboot

```
Status_t    SP7_Reboot          (     void );
```

Reinstall all register settings, which were stored during the first start-up, and reload all default algorithm constants. Starts demodulator.

3.5.2. SP7_Reset

```
Status_t    SP7_Reset           ( void );
```

Reinstalls all register settings, does not reinitialize algorithm constants, such that user patches remain active. Restarts algorithm.

3.5.3. SP7_SysReset

```
Status_t    SP7_SysReset        ( void );
```

First resets all internal hardware, then behaves like *SP7_Reset*.

3.5.4. SP7_Restart

```
Status_t    SP7_Restart         ( void );
```

Just restart algorithm, with current register settings and algorithm constants.

3.5.5. SP7_Halt

```
Status_t    SP7_Halt            ( void );
```

Halts the current state of the DRX 8872C. This is a useful function when debugging your application. The DRX 8872C remains executing commands.

3.5.6. SP7_Nop

```
Status_t    SP7_Nop             ( void );
```

Do nothing, but generate an interrupt on completion.

The next two functions enable different algorithm constants to be programmed to influence the behavior of the algorithm. Also the state of all microcode variables can be queried. Reading and writing microcode variables is done “atomically”.

3.5.7. SP7_Wreg

```
Status_t    SP7_Wreg            (     u16_t       Reg,
                                      u16_t       Data
                                );
```

Variable	Type	Description
Reg	u16_t	Address of register that needs to be written.
Data	u16_t	New value for the register.

3.5.8. SP7_Rreg

```
Status_t    SP7_Rreg            (     u16_t       Reg,
                                      pu16_t      Data
                                );
```

Variable	Type	Description
Reg	u16_t	Address of register that needs to be read.
Data	pu16_t	Returns value of the register.

3.5.9. SP7_Wvar

```
Status_t    SP7_Wvar           (     u16_t       Adr,
                                     u16_t       Wnr,
                                     pu16_t      Wdata
                               );
```

Variable	Type	Description
Adr	u16_t	Address of (processor) variable that needs to be written.
Wnr	u16_t	Number of words that must be written.
Wdata	pu16_t	New value for the variable (array).

3.5.10.SP7_Rvar

```
Status_t    SP7_Rvar           (     u16_t       Adr,
                                     u16_t       Rnr,
                                     pu16_t      Rdata
                               );
```

Variable	Type	Description
Adr	u16_t	Address of (processor) variable that needs to be read.
Rnr	u16_t	Number of words that must be read.
Rdata	pu16_t	Returns value of the variable (array).

3.6. Interrupts and Events

The DRX 8872C has an IRQN output which can generate an interrupt to the host to trigger an event. Two types of interrupts are reported back to the host: “Command Completion” interrupt, and “Algorithm Event” interrupt. The API uses the “Command Completion” interrupt internally. The “Algorithm Event” interrupt is of special interest to the application.

3.6.1. SP7_ReadIrq

```
Status_t    SP7_ReadIrq        ( void );
```

Reads the interrupt status register, such that new interrupts may occur.

On any occasion, the event register may be read to see if the DRX 8872C algorithm generated an event. If an event occurs, an interrupt is also generated. Thus if the interrupt line is used, it makes sense to only read the event register on an interrupt.

3.6.2. SP7_PollEvent

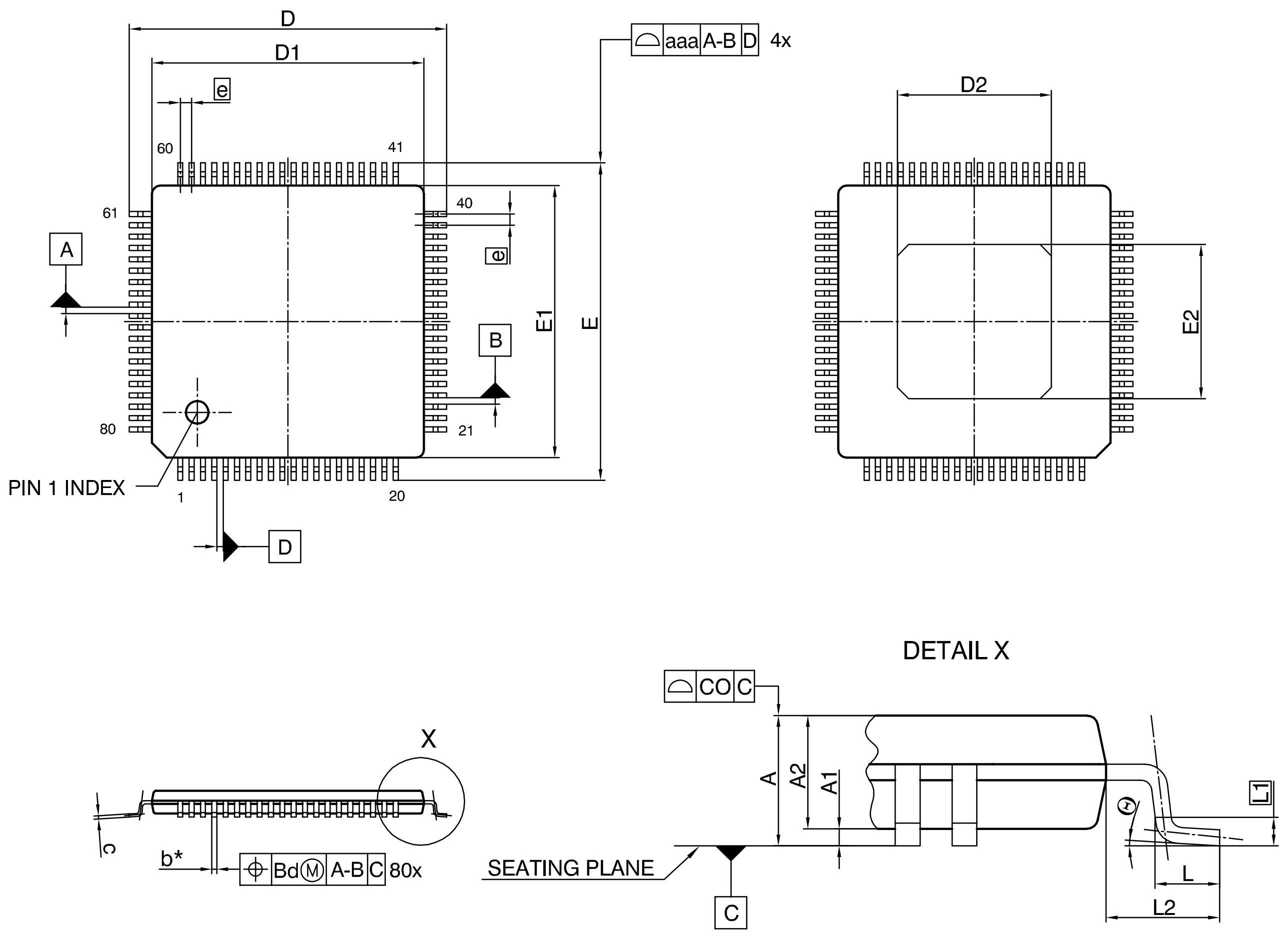
```
Status_t    SP7_PollEvent       (     pu16_t      OccurredEvents );
```

Reads the event register and sets bits in OccurredEvents according to the events that occurred since the last read.

Variable	Type	Description
OccurredEvents	pu16_t	The following event bits may be set: SP7_EV_ERR Algorithm restarted (lock lost). SP7_EV_SIG Algorithm detected a DVB-T signal, lock will follow soon. SP7_EV_LCK Algorithm acquired a lock on the DVB-T signal and is demodulating now. SP7_EV_VBER A new Pre-Viterbi BER measurement is ready to be read. SP7_EV_RBER A new Post-Viterbi BER measurement (or packet error count) is ready to be read. SP7_EV_SN A new SN and MER measurement is ready to be read. SP7_EV_TPS A new TPS frame is ready to be read.

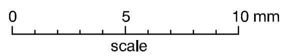
4. Specifications

4.1. Outline Dimensions



UNIT	L	L1	L2	Θ
mm	0.75 0.45	0.25	1.0	0°-7°

* does not include dambar protrusion of 0.08 max. per side



UNIT	A	A1	A2	aaa	b	Bd	c	CO	D	D1	D2	E	E1	E2	e
mm	1.2 max	0.15 0.05	1.05 0.95	0.2	0.27 0.17	0.08	0.20 0.09	0.05	14.0	12.0	7.0 6.6	14.0	12.0	7.0 6.6	0.5

JEDEC STANDARD		ANSI	ISSUE DATE YY-MM-DD	DRAWING-NO.	SPZG-NO.
ISSUE	ITEM NO.				
C	MS-026		03-08-14	06638.0001.4	SPZG001038_001_02

Fig. 4–1:
PTQFP80-1: Plastic **T**hin **Q**uad **F**lat **P**ackage, **80** leads, 12 × 12 × 1.0 mm^3, exposed die pad
Ordering code: PL

Weight approximately 1.0 g

4.2. Pin Connections and Short Descriptions

Pin No.	Pin Name	Type	Connection (If not used)	Short Description
1	VDDL	P		Digital core supply 2.5 volt
2	VSS	GND		Digital ground
3	VSS	GND		Digital ground
4	ASEL	I		Serial interface address select
5	VSS	GND		Digital ground
6	AGC	O		Automatic gain output
7	VSSA	GND		Analog ground
8	VREF	P		Bias circuitry reference voltage decoupling pin
9	INM	I		Symmetrical IF or Base band input
10	INP	I		Symmetrical IF or Base band input
11	VSSA	GND		Analog ground
12	VDDA	P		Analog supply 2.5V
13	VRM	P		External middle rail reference
14	VDDA	P		Analog supply 2.5V
15	VSSA	GND		Analog ground
16	VRH	P		High reference Voltage
17	VRL	P		Low reference Voltage
18	VSSA	GND		Analog ground
19	VSSA	GND		Analog ground
20	VDDH	P		Digital IO supply 3.3V
21	SCL	O		Clock line for serial protocol
22	SDA	I/O		Data line for serial protocol
23	VSS	GND		Digital ground
24	TCLK	I	10 kΩ pull-up to VDDH	JTAG test clock
25	TDI	I	10 kΩ pull-up to VDDH	JTAG test data in
26	TMS	I	10 kΩ pull-up to VDDH	JTAG test mode select
27	TDO	O		JTAG test data out

Pin No.	Pin Name	Type	Connection (If not used)	Short Description
28	VDDL	P		Core supply 2.5V
29	VSS	GND		Digital ground
30	VDDH	P		Digital IO supply 3.3V
31	VSS	GND		Digital ground
32	VSS	GND		Digital ground
33	VSS	GND		Digital ground
34	VSS	GND		Digital ground
35	IRQN	O		Interrupt to host (active low)
36	VDDL	P		Digital core supply 2.5V
37	VSS	GND		Digital ground
38	SCL2	O		Secondary serial interface clock
39	MERR	O		MPEG packet error flag
40	MSTRT	O		Frame start flag
41	VDDL	P		Digital core supply 2.5V
42	VDDH	P		Digital IO supply 3.3V
43	MVAL	O		MPEG2 data valid signal
44	MCLK	O		MPEG2 clock
45	MD_7	O		MPEG2 data output
46	VSS	GND		Digital ground
47	MD_6	O		MPEG2 data output
48	MD_5	O		MPEG2 data output
49	MD_4	O		MPEG2 data output
50	VDDH	P		Digital IO supply 3.3V
51	VSS	GND		Digital ground
52	MD_3	O		MPEG2 data output
53	MD_2	O		MPEG2 data output
54	MD_1	O		MPEG2 data output
55	MD_0	O		MPEG2 data output/ MPEG serial data output
56	OFDM_LCK	O		Lock signal indicating valid OFDM signal
57	VSS	GND		Digital ground
58	VDDH	P		Digital IO supply 3.3V

Pin No.	Pin Name	Type	Connection (If not used)	Short Description
59	FEC_LCK	O		Lock signal indicating FEC lock
60	NC		LV	Should be left open
61	VDDL	P		Digital core supply 2.5V
62	I2C_EN	O		Secondary serial interface enable line
63	VSS	GND		Digital ground
64	VSS	GND		Digital ground
65	VSS	GND		Digital ground
66	VSS	GND		Digital ground
67	VSS	GND		Digital ground
68	VSS	GND		Digital ground
69	VDDH	P		Digital IO supply 3.3V
70	VSS	GND		Digital ground
71	XO	O		Crystal oscillator output
72	NC			Internally not connected
73	XI	I		Crystal oscillator input
74	VSS	GND		Digital ground
75	XPD	I		Power down mode low = oscillator active
76	VDDL	P		Digital core supply 2.5V
77	VSS	GND		Digital ground
78	IREFSEL	I		Internal voltage reference select high = enable
79	RST	I		Reset signal (active low)
80	VSS	GND		Digital ground

4.3. Pin Configurations

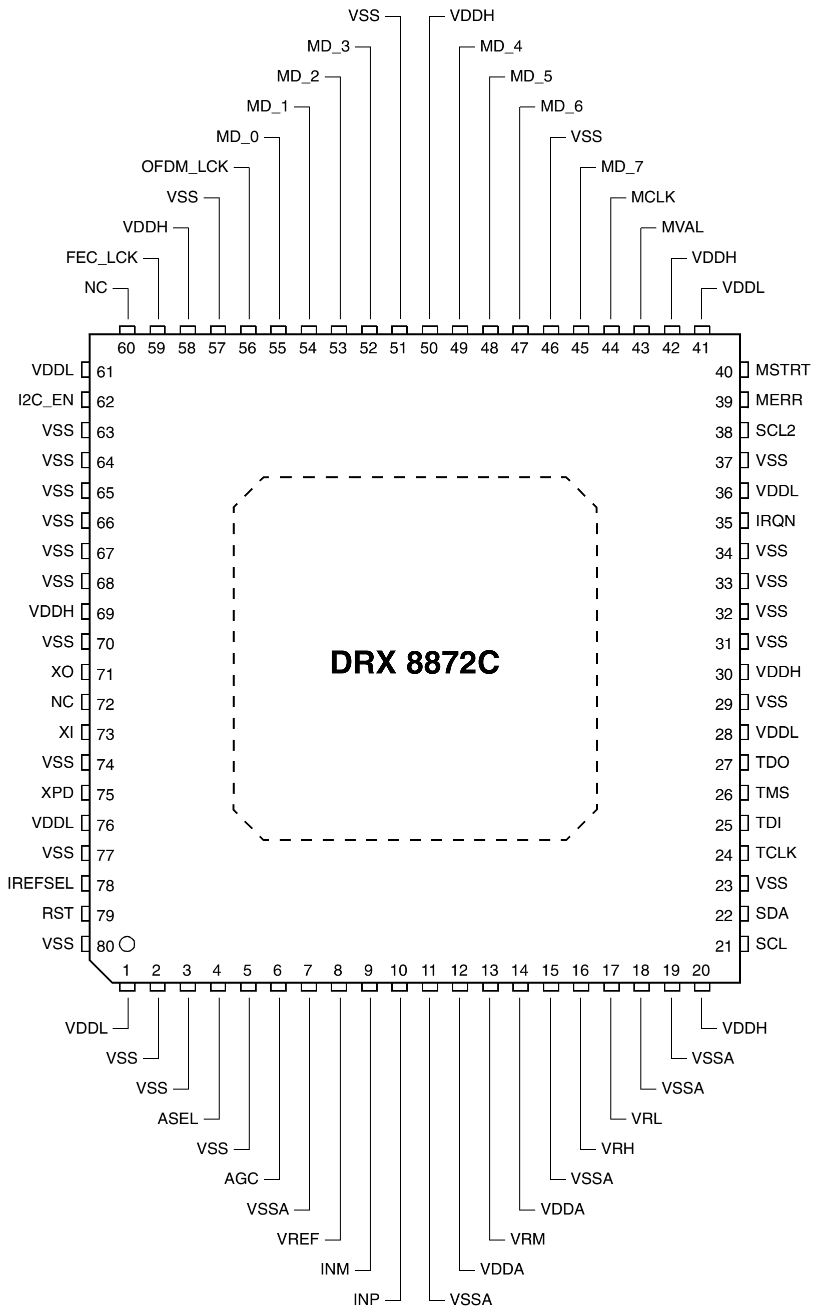


Fig. 4–1: 80-pin PTQFP package

4.4. Electrical Characteristics

Abbreviations

tbd = to be defined
vacant = not applicable
positive current values mean current flowing into the chip

4.4.1. Absolute Maximum Ratings

Stresses beyond those listed in the "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these conditions is not implied. Exposure to absolute maximum rating conditions for extended periods will affect device reliability.

This device contains circuitry to protect the inputs and outputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than absolute maximum-rated voltages to this high-impedance circuit.

All voltages listed are referenced to ground (list voltages = 0 V) except where noted.

All GND pins must be connected to a low-resistive ground plane close to the IC.

Table 4–1: Absolute Maximum Ratings

Symbol	Parameter	Pin Name	Limit Values		Unit
			Min.	Max.	
T_A	Ambient Operating Temperature	–	0	70[1)]	°C
T_S	Storage Temperature	–	–40	125	°C
P_{MAX}	Maximum Power Dissipation	–	–	1.1	W
V_{DDA}	Analog supply	VDDA	–0.3	3.0	V
V_{DDL}	Digital core supply	VDDL	–0.3	3.0	V
V_{DDH}	Digital IO supply	VDDH	–0.3	3.6	V
ΔV_{SUP}	Voltage differences within supply domains (V_{DDA} - V_{DDL})	–	–0.2	0.2	V
$V_{I\ analgoue}$	Analog Input Voltage	INM; INP; AGC; VRH; VRL	–0.3	V_{DDA} + 0.3	V
$V_{I\ digital}$	Digital Input Voltage	ASEL; SDA; SCL; TDI; TMS; IRQN; XI, XPD, IREFSEL, RST	–0.3	V_{DDH} + 0.3	V

1) A thermally-optimized board layout is recommended; refer to the application note "Surface Mount Assembly of Exposed-Pad QFP packages".

4.4.2. Recommended Operating Conditions

Functional operation of the device beyond those indicated in the “Recommended Operating Conditions/Characteristics” is not implied and may result in unpredictable behavior, reduce reliability and lifetime of the device.

All voltages listed are referenced to ground (list voltages = 0 V) except where noted.

All GND pins must be connected to a low-resistive ground plane close to the IC.

Do not insert the device into a live socket. Instead, apply power by switching on the external power supply. For power up/down sequences, see the instructions in section xxx of this document.

4.4.2.1. General Recommended Operating Conditions

Symbol	Parameter	Pin Name	Limit Values			Unit
			Min.	Typ.	Max.	
T_A	Ambient Operating Temperature	–	0		70[1)]	°C
P_{MAX}	Maximum Power Dissipation	–	–	–	1.1	W
V_{DDA1}	Analog Supply	VDDA	2.3	2.5	2.7	V
V_{DDL}	Digital Core Supply	VDDL	2.3	2.5	2.7	V
V_{DDH}	Digital IO Supply	VDDH	3.0	3.3	3.6	V

[1)] A thermally-optimized board layout is recommended; refer to the application note “Surface Mount Assembly of Exposed-Pad QFP packages”

4.4.3. Characteristics

4.4.3.1. DC Electrical Characteristics

Symbol	Parameter	Pin Name	Min.	Typ.	Max.	Unit	Test Conditions
V_{DDL}	Digital Core Supply	VDDL	2.3	2.5	2.7	V	
V_{DDA}	Analog Supply	VDDA	2.3	2.5	2.7	V	
V_{DDH}	Digital IO Supply	VDDH	3.0	3.3	3.6	V	
I_{DDL}	Digital Core Supply Current	VDDL	–	380	420	mA	
I_{DDA}	Analog Supply Current	VDDA	–	15	25	mA	
I_{DDH}	Digital Io Supply Current	VDDH	–	32	40	mA	

4.4.3.2. Temperature Ratings

Symbol	Parameter	Pin Name	Min.	Typ.	Max.	Unit	Test Conditions
T_J	Junction Temperature	–	–	–	125	°C	
T_A	Ambient Temperature Range	–	0	–	70	°C	
T_S	Storage Temperature Range	–	–40	–	125	°C	
T_C	Case Temperature Range	–	–	–	110	°C	

4.4.3.3. ADC Parameters

Symbol	Parameter	Pin Name	Min.	Typ.	Max.	Unit	Test Conditions
V_{DDA}	ADC Power Supply	VDDA	2.3	2.5	2.7	V	
P_{analog}	Analog Power Consumption		–	50	70	mW	
(INM-INP)	Input Voltage Swing	INM, INP	±0.5	±1.2	–	V	
V_{RH}	High Voltage Reference	VRH	–	1.6	1.7	V	
V_{RL}	Low Voltage Reference	VRL	0.7	0.85	–	V	
R_{in}	Input Impedance	INM, INP	–	200	–	Ω	
C_{in}	Input Capacitance	INM, INP	–	2.5	–	pF	
V_{CM}	Common Mode	VRM	1.0	1.2	1.35	V	

5. Appendix A – API Data Types

For ease of porting the library across different operating systems and platforms, basic data types have been defined for 8-bit, 16-bit and 32-bit signed and unsigned integers. These data type definitions are valid for most 32-bit platforms. When porting the API to a 16-bit or 8-bit platform, you may need to redefine a few of these data types.

Additionally, chip-specific data types are defined.

All data types are defined in the file “SP7_Type.h”

5.1. Basic Data Types

(p)u8_t	(pointer to) unsigned 8-bit integer value: 0…255
(p)s8_t	(pointer to) signed 8-bit integer value: –128…127
(p)u16_t	(pointer to) unsigned 16-bit integer value: 0…65,535
(p)s16_t	(pointer to) signed 16-bit integer value: -32,768…32,767
(p)u32_t	(pointer to) unsigned 32-bit integer value: 0…4,294,967,295
(p)s32_t	(pointer to) signed 32-bit integer value: -2,147,483,648…2,147,483,647
(p)Bool_t	(pointer to) Boolean value: TRUE, FALSE

5.2. Chip-specific Data Types

(p)AGC_t	(pointer to) AGC control data-type value: AGC_OFF, AGC_FAST, AGC_SLOW, AGC_INV
(p)Bandwidth_t	(pointer to) bandwidth data type value: AUTO, BANDWIDTH_8MHZ, BANDWIDTH_7MHZ, BANDWIDTH_6MHZ
(p)Cls_t	(pointer to) channel classification data-type value: AUTO, CLS_GAUSS, CLS_HEAVYGAUSS, CLS_COCHANNEL, CLS_STATIC, CLS_MOVING, CLS_ZERODB
(p)Const_t	(pointer to) constellation data-type value: AUTO, CONST_QPSK, CONST_QAM16, CONST_QAM64
(p)Guard_t	(pointer to) guard data-type value: AUTO, GUARD_32, GUARD_16, GUARD_8, GUARD_4
(p)Hier_t	(pointer to) hierarcy data-type value: AUTO, HIER_NONHIER, HIER_ALPHA_1, HIER_ALPHA_2, HIER_ALPHA_4
(p)LCK_t	(pointer to) LCK pin usage data-type value: LCK_UIO_0, LCK_UIO_1, LCK_INDICATOR, LCK_SAW_8, LCK_SAW_7
(p)Mirror_t	(pointer to) mirrored signal data-type value: AUTO, NORMAL, MIRROR
(p)Mode_t	(pointer to) mode data-type value: AUTO, MODE_2K, MODE_8K
(p)Prior_t	(pointer to) priority data-type value: PRIOR_HIGH, PRIOR_LOW
(p)Rate_t	(pointer to) code rate data-type value: AUTO, RATE_1_2, RATE_2_3, RATE_3_4, RATE_5_6, RATE_7_8
(p)Status_t	(pointer to) NIM function call result status value: STS_OK, STS_BUSY, STS_INVALID_ARG, STS_ERROR
(p)TpsFrame_t	(pointer to) TPS frame number data-type value: TPS_FRAME_1, TPS_FRAME_2, TPS_FRAME_3, TPS_FRAME_4

6. Appendix B – API Required Platform Functions

The API interfaces with the DRX 8872C via a serial protocol. The implementation of the serial protocol is platform dependent.

If your platform is a PC running Microsoft Windows 98, 2000, NT, or XP you can use serial interface functions as supplied that make use of the Micronas proprietary serial driver that implements serial access via a parallel port. Refer to section 6.1.

For other platforms, you will have to write your own serial access functionality. Refer to Section 6.2.

6.1. PC Platforms Running Windows 98, NT, 2000, and XP

If you want to use the Micronas serial driver for serial access via a parallel port, do the following:

1. Install the Signal Spyder application (as supplied on the CD). Refer to Signal Spyder manual for instructions.

   In file “SP7_Conf.h”, make sure that the following definition is enabled:

   ```
   #define SP7_I2C_SPASE
   ```

   By default, this line is commented out.
2. Compile the API and the supplied file “i2ctest.c” and link them into an executable “i2ctest.exe”.
3. Make sure that the file “I2CECP.DLL” (as provided in the Signal Spyder distribution) is in the search path.
4. Test the serial access by running “i2ctest.exe”.

6.2. Other platforms

The API assumes that the following two functions are made available to read and write registers inside the DRX 8872C:

```
Status_t  SP7_I2C_Write(  u16_t  NrOfWbytes,  pu8_t
Wdata  )
Status_t  SP7_I2C_Read  (  u16_t  NrOfWBytes,  pu8_t
Wdata, u16_t NrOfRbytes, pu8_t Rdata  )
```

The *SP7_I2C_Write* and *SP7_I2C_Read* functions must be implemented using the available serial functionality of the specific platform.

The functions must be created in the file “SP7_I2c.c”’. Below, the requirements for these functions are described in detail.

After these serial functions have been implemented, use the supplied file “i2ctest.c” to verify the implementation.

6.2.1. SP7_I2C_Write

Writing data to the DRX8872C involves the following steps:

1. Generate I^2C start condition.
2. Write the bytes pointed to by `Wdata`. *Wdata* will include the device address (0xE0 or 0xE2 for the DRX 8872C and 0xC0 for tuner part). The number of bytes that must be written is `NrOfWBytes`.
3. Generate I^2C stop condition

   Or graphically:

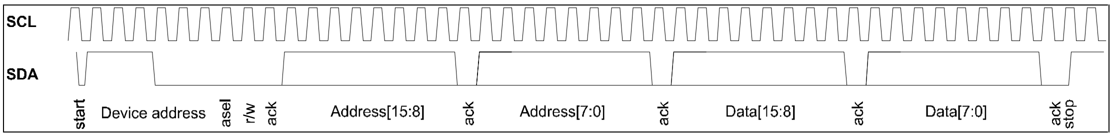


Fig. 6–1: I^2C_Write timing (in this case NrOfWbytes=5).

6.2.2. SP7_I2C_Read

Reading data from the DRX8872C involves the following steps:

1. Generate I^2C start condition.
2. Write the bytes pointed to by `Wdata`. The number of bytes that must be written is `NrOfWBytes`. This is used to setup the device address and the internal address of the register that is going to be programmed. Before the actual read operation starts, the device address needs to be setup again. Therefore the last byte of the `Wdata` array needs to be preceded by a start operation as indicated in the picture below.
3. Read `NrOfRBytes` consecutive data-bytes (minimum of 2 bytes) into the memory location pointed to by `Rdata`.
4. Generate I^2C stop condition.

   Or graphically:

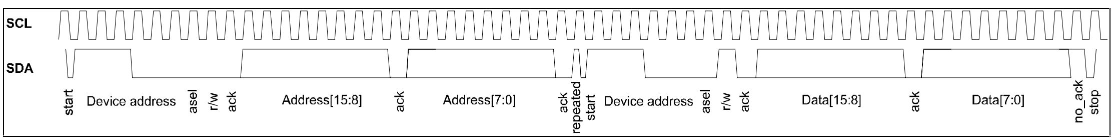


Fig. 6–2: I^2C_Read timing (in this case NrOfWbytes=4, NrOfRbytes=2).

6.2.3. I2ctest.c

The API distribution includes the source file “i2ctest.c”.

The program “I2Ctest” can be used to test the functionality of your “SP7_I2c.c” implementation. Simply compile and link these two programs for your platform and then run the code. The program will return 0 (zero) if the test was successful, otherwise –1 if any error occurred. Additionally, if you define VERBOSE during compilation, the program will also show results using “printf” calls.

In this file, the I^2C address is set at 0xE0. Depending on your DRX8872C configuration, you may need to change this I^2C address to 0xE2.

```
/******************************************************************************
* FILENAME: i2ctest.c
*
* DESCRIPTION:
* Test the customer provided SP7_I2C_Read and SP7_I2C_Write functions
*
* This program does the following:
*
* - Switch off system controller
* - Write test pattern in SC_INFO0 register
* - Write different test pattern in SC_INFO1 register
* - Read from SC_INFO0 register
* - Check if read-back value from SC_INFO0 equals written value
*
* By default, it uses I2C address 0xE0. To use 0xE2 redefine I2C_ADDR in i2ctest.c.
*
* By default, it uses stdout to print the test results.
* This can be switched off by removing the line:
* #define VERBOSE
*
* This program only uses the SP7_I2C_Read and SP7_I2C_Write functions in the file SP7_I2C.c.
*
* USAGE:
* Link with SP7_I2C.c to separate executable
* Define VERBOSE to output test results to stdout
*
* NOTES:
* $(c) 2003 Micronas GmbH. All rights reserved.
*
* This software and related documentation (the 'Software') are intellectual
* property owned by Micronas and are copyright of Micronas, unless specifically
* noted otherwise.
*
* Any use of the Software is permitted only pursuant to the terms of the
* license agreement, if any, which accompanies, is included with or applicable
* to the Software ('License Agreement') or upon express written consent of
* Micronas. Any copying, reproduction or redistribution of the Software in
* whole or in part by any means not in accordance with the License Agreement
* or as agreed in writing by Micronas is expressly prohibited.
*
* THE SOFTWARE IS WARRANTED, IF AT ALL, ONLY ACCORDING TO THE TERMS OF THE
* LICENSE AGREEMENT. EXCEPT AS WARRANTED IN THE LICENSE AGREEMENT THE SOFTWARE
* IS DELIVERED 'AS IS' AND MICRONAS HEREBY DISCLAIMS ALL WARRANTIES AND
* CONDITIONS WITH REGARD TO THE SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES
* AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIT
* ENJOYMENT, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL
* PROPERTY OR OTHER RIGHTS WHICH MAY RESULT FROM THE USE OR THE INABILITY
* TO USE THE SOFTWARE.
*
* IN NO EVENT SHALL MICRONAS BE LIABLE FOR INDIRECT, INCIDENTAL, CONSEQUENTIAL,
* PUNITIVE, SPECIAL OR OTHER DAMAGES WHATSOEVER INCLUDING WITHOUT LIMITATION,
* DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS
* INFORMATION, AND THE LIKE, ARISING OUT OF OR RELATING TO THE USE OF OR THE
* INABILITY TO USE THE SOFTWARE, EVEN IF MICRONAS HAS BEEN ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGES, EXCEPT PERSONAL INJURY OR DEATH RESULTING FROM
* MICRONAS' NEGLIGENCE.                                                       $
*
******************************************************************************/

#include "SP7.h"
#include "SP7_Regs.h"
```

```
/* Change this I2C address to 0xE2 if ASEL pin of demodulator chip is high */
#define I2C_ADDR            0xE0

/* Remove next line if printf's must be suppressed */
#define VERBOSE

#ifdef VERBOSE
#include <stdio.h>
#endif

/*****************************************************************************
* Write a value to a register
*****************************************************************************/
Status_t WriteReg (u16_t Reg, u16_t Data)
{
   u8_t Wdata[5];
   Status_t rc;

#ifdef VERBOSE
    printf("Writing: I2C_addr: 0x%04x, Reg: 0x%04x <== 0x%04x ", I2C_ADDR,  Reg, Data);
#endif

   Wdata[0] = (u8_t) I2C_ADDR;
   Wdata[1] = (u8_t)((Reg  >> 8) & 0xff);
   Wdata[2] = (u8_t)((Reg  >> 0) & 0xff);
   Wdata[3] = (u8_t)((Data >> 8) & 0xff);
   Wdata[4] = (u8_t)((Data >> 0) & 0xff);

   rc = SP7_I2C_Write(5, Wdata);

   if (rc == STS_OK) {
#ifdef VERBOSE
        printf("OK\n");
#endif
   } else {
#ifdef VERBOSE
        printf("FAILED\n");
#endif
   }

   return rc;
}

/*****************************************************************************
* Read from a register
*****************************************************************************/
Status_t ReadReg (u16_t Reg, pu16_t Data)
{
   u8_t     Wdata[4];
   u8_t     Rdata[2];
   Status_t rc;

#ifdef VERBOSE
    printf("Reading: I2C_addr: 0x%04x, Reg: 0x%04x ", I2C_ADDR,  Reg);
#endif

   Wdata[0] = (u8_t)(I2C_ADDR);
   Wdata[1] = (u8_t)((Reg  >> 8) & 0xff);
   Wdata[2] = (u8_t)((Reg  >> 0) & 0xff);

   /* I2C repeated start in between by  */
   Wdata[3] = (u8_t)(I2C_ADDR | 0x01);

   rc = SP7_I2C_Read(4, Wdata, 2, Rdata);

   *Data = (u16_t)((Rdata[0] << 8) | Rdata[1]);

#ifdef VERBOSE
   printf("==> 0x%04x ", *Data);
   if (rc == STS_OK) {
        printf("OK\n");
   } else {
        printf("FAILED\n");
   }
#endif

   return rc;
}
```

```
/******************************************************************************
* Main test program
******************************************************************************/
int main(void)
{
    u16_t value;
    int rc = 0;

    /* initialize I2C functionality */
    SP7_I2C_Init();

    if (WriteReg(SP7_SC_MV_MODE, 0) != STS_OK) {
        rc = -1;
    }

    /* write something in SP7_SC_INFO0 */
    if (WriteReg(SP7_SC_INFO0, 0x0aaa) != STS_OK) {
        rc = -1;
    }

    /* write something else in SP7_SC_INFO1 */
    if (WriteReg(SP7_SC_INFO1, 0x0555) != STS_OK) {
        rc = -1;
    }

    /* read SP7_SC_INFO0 */
    if (ReadReg(SP7_SC_INFO0, &value) != STS_OK) {
        rc = -1;
    }

    if (value != 0x0aaa) {
        rc = -1;
#ifdef VERBOSE
        printf("Read back check FAILED\n");
#endif
    } else {
#ifdef VERBOSE
        printf("Read back check PASSED\n");
#endif
    }

#ifdef VERBOSE
    if (rc == -1) {
        printf("\nI2C test FAILED\n");
    } else {
        printf("\nI2C test PASSED\n");
    }
#endif

    /* Terminate I2C functionality  */
    SP7_I2C_Term();

    return rc;
}
```

7. Data Sheet History

1. Data Sheet: “DRX 8872C COFDM Demodulator/ FEC”, May 28, 2003, 6251-618-1DS. First release of the data sheet.
2. Data Sheet: “DRX 8872C COFDM Demodulator/ FEC”, Feb. 18, 2004, 6251-618-2DS. Second release of the data sheet.

Micronas GmbH
Hans-Bunte-Strasse 19
D-79108 Freiburg (Germany)
P.O. Box 840
D-79008 Freiburg (Germany)
Tel. +49-761-517-0
Fax +49-761-517-2174
E-mail: docservice@micronas.com
Internet: www.micronas.com

Printed in Germany
Order No. 6251-618-2DS