APS022 APPLICATION NOTE

DEBUGGING DW1000 BASED PRODUCTS AND SYSTEMS

Help with debugging DW1000 based applications

Version 1.3

This document is subject to change without notice

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Scope

This document is intended to help embedded software developers and firmware engineers debug their software as they develop applications and drivers to control the DW1000.

1.2 Pre-requisites

The developer should be familiar with the DW1000 Data Sheet [1] and the DW1000 User Manual [2].

1.3 References and sources

The software code snippets referred to in this document are taken from the source code supplied with the "DW1000 Application Programming Interface with application examples"[1] and built with the CooCox IDE[2].

The hardware platform referred to in this document is the EVB1000 platform.

1.4 Document overview

- Chapter 2 reviews the power up sequence of DW1000
- Chapter 3 describes common SPI problems
- Chapter 4 shows how to use system events and state registers to help in diagnosing any issues
- Chapter 5 covers transmission
- Chapter 6 reviews reception
- Chapter 7 provides help on configuring and using DW1000 sleep / low power states

[1] May be downloaded from www.decawave.com/support/software
[2] Version 1.7.8 with GNU Tools ARM for Embedded toolchain which may be downloaded by following this link https://www.coocox.org/download/Tools/CoIDE-1.7.8.exe

2 POWER-UP SEQUENCE

As the DW1000 IC is powered up, various pins can be probed to verify that the device is operating correctly and that at the end of the power up sequence it is in its IDLE state. The DW1000 Data Sheet [1] shows the power-up sequence and the status of the main DW1000 pins.

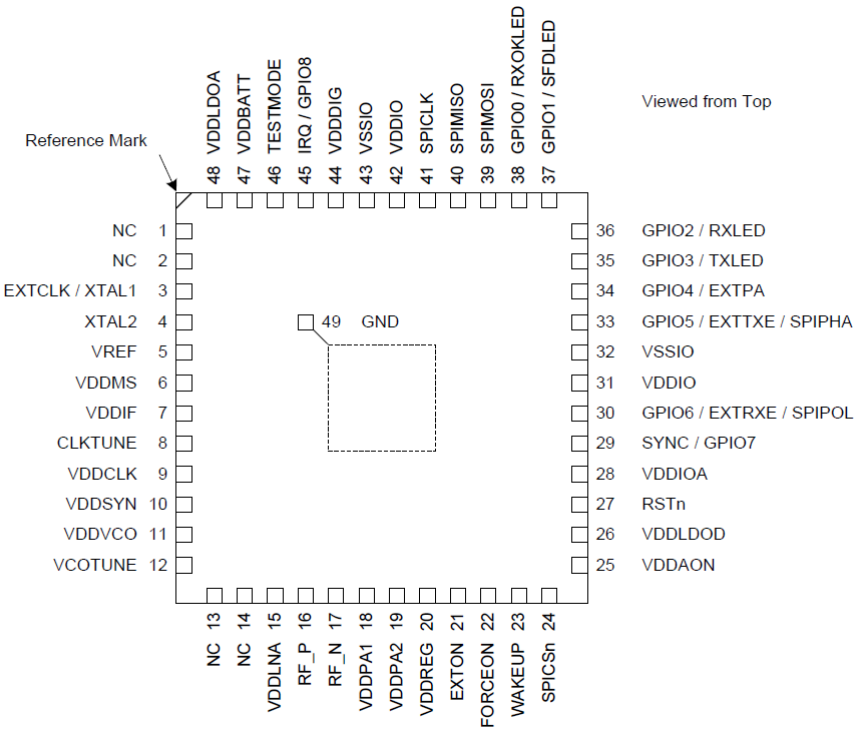

Figure 1: DW1000 pin assignments

1. Firstly the 3.3 V supplies should be switched on (as shown in Figure 2)
2. Soon after the voltage on the supply pins reaches V_{on} (2.0 V nominal value – see DW1000 Data Sheet [1]) the EXTON pin will go high.
3. In parallel the XTAL will start oscillating, and will reach the desired frequency of 38.4 MHz after T_{osc_on}.
4. Then VDDREG and VDDDIG will go high followed by RSTn.
5. The transition of RSTn from low to high indicates that the DW1000 is in the INIT state. The time taken for this transition depends on whether the DW1000 is powering up or is being woken up from the DEEPSLEEP state.

    a. In the WAKEUP case, it can take up to 40 µs for the transition from the INIT state to the IDLE state; this is due to the download of the always on (AON) array after WAKEUP. The AON block contains a low-power configuration array that retains the on-chip register configuration and remains powered-up provided power (from the battery, for example) is supplied to the DW1000 via the VDDAON pin.
    b. In the case of power-up, the DW1000 transitions from the INIT to the IDLE state after approximately 5 µs (the time it takes the on-chip PLL to lock).

6. Once in IDLE mode the SPI can operate at up to 20 MHz rate. The VREF voltage should be 1.12 V.
7. Other supply pins should also be checked when in the IDLE state (VDDDIG, VDDMS, VDDCLK etc.) to ensure they are at the correct voltage. Refer to [1].

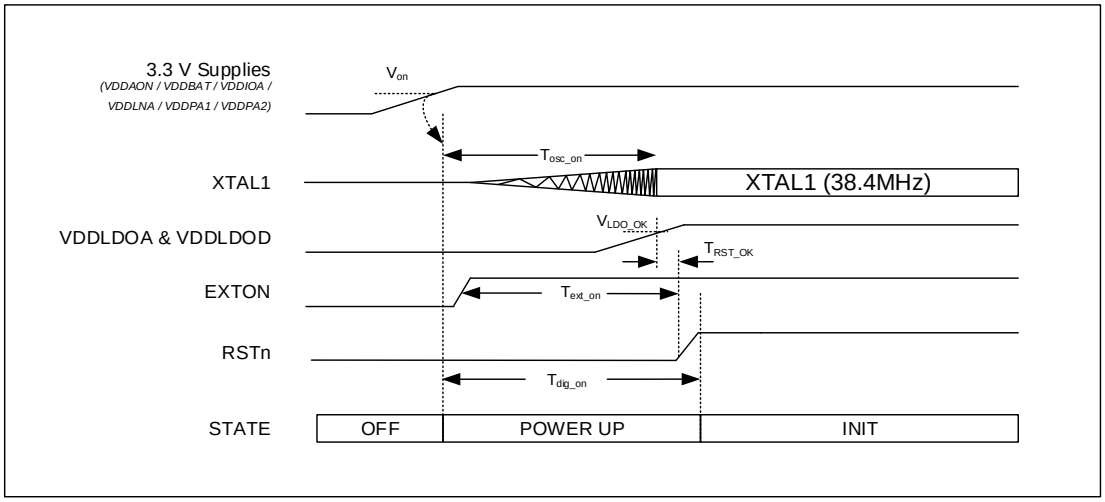

Figure 2: DW1000 power up sequence

3 DW1000 INTERFACES (SPI AND GPIOS)

3.1 DW1000 SPI interface

The DW1000 host communications interface is a slave-only SPI. Both clock polarities (SPIPOL=0/1) and phases (SPIPHA=0/1) are supported, as defined in the SPI Mode Configuration table in the DW1000 Data Sheet [1].

On any new platform the SPI interface functions should be checked so that both reading and writing DW1000 registers works correctly. The data written to the device is not checked / verified by the device so the developer should make sure the read and write functions operate correctly. The programmable SFD register file, USR_SFD at address 0x21 may be used for this and is 401 bytes long.

```
uint8 dataA[20] = { some test data };
uint8 dataB[20];

dwt_writetodevice(0x21, 0, 20, &dataA[0]);
dwt_readfromdevice(0x21, 0 , 20, &dataB[0]);

if dataA matches dataB then the SPI read and write functions are working correctly.
```

3.2 Low and high SPI rate

The DW1000 requires that some of the register read and write operations are done at an SPI rate of < 3 MHz, otherwise a rate of up to 20 MHz can be used. This is clearly specified in the DW1000 User Manual [2] and DW1000 Software API Guide [3].

If an SPI rate faster than 3 MHz is used for the low rate read and write operations then the chip will not operate correctly and a hard reset will be needed to re-enable communications.

3.3 Read device ID

After the DW1000 is powered up, calling an SPI read of register DEV_ID (32-bits) should return 0xDECA0130 if the SPI is operating correctly (MSB is 0xDE, LSB is 0x30).

If the device returns 0xFFFFFFFF then: -

- it could be in DEEPSLEEP in which case it needs to be woken up before the SPI read / write will operate correctly; or
- the SPI may not have been configured correctly.

Please check the SPI pin connections and the controller's SPI configuration.

If the device returns some other value e.g. 0xBD940260: -

- the SPI mode is probably not set correctly (the controller's SPI mode does not match the DW1000 SPI mode); or
- the data read back is shifted. Check the controller's SPI configuration.

```
uint32 devID = dwt_readdevid();
devID will be 0xDECA0130 if SPI read function is working correctly.
```

3.4 SPI configuration modes

As described in the DW1000 Data Sheet [1], GPIO 5 and 6 are used for SPI mode configuration. They

are sampled (latched) on the rising edge of the RSTn pin to determine the SPI mode.

They are internally pulled low to configure a default SPI mode of 0 without the use of external components.

If a mode other than 0 is required then they should be pulled up using an external resistor of value no greater than 10 kΩ connected to the VDDIO output supply.

Note that GPIO 5 and 6 are connected to SW3 on the EVB1000 (switches 3 and 4), and that these switches may be used to configure desired SPI modes.

3.5 DW1000 GPIO5 and GPIO6

The DW1000 has a number of GPIOs. These are defined in the DW1000 Data Sheet [1]. Of particular interest are GPIOs 5 and 6. As well as selecting the SPI mode, as discussed above, these can be configured to output TX and RX states as follows: -

- GPIO 6 may be configured for use as EXTRXE (External Receiver Enable). This pin goes high when the DW1000 is in receive mode (receiver is active).
- GPIO 5 may be configured for use as EXTTXE (External Transmit Enable). This pin goes high when the DW1000 is actively transmitting data.

If delayed transmission or reception is used then the signal will stay low until the delay has passed and the DW1000 changes from IDLE to TX or RX mode. A call to API function `dwt_setlnapamode(1,1)` configures these GPIO 5 and 6 functions.

3.6 TX and RX states on an oscilloscope

It is very informative to look at the TX and RX "on" times to check that the device is operating as expected. For example if a device is configured to transmit a frame and then enter the receive state automatically (after a particular delay), both the TX and RX state signals can be viewed on an oscilloscope and the TX and RX timings can be confirmed.

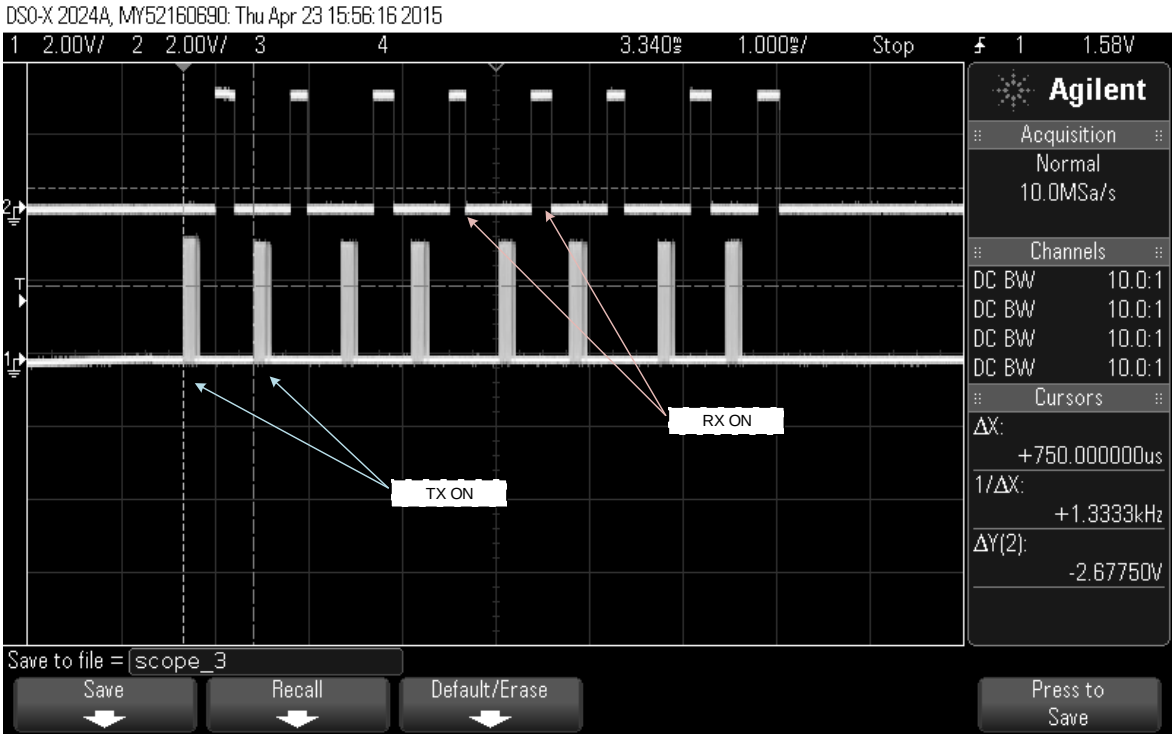

Figure 3: TX and RX states on the oscilloscope

Figure 3 shows an example of this. Here channel 1 on the oscilloscope (lower trace in Figure 3) is used to monitor the TX state and channel 2 (upper trace in Figure 3) the RX state. The device initially transmits a frame then turns on its receiver. After the receive finishes the device transmits the frame again and so on.

Monitoring these signals in this way is very useful when debugging, for example, a multi-device environment, where some devices are transmitting and others are expected to be in receive mode.

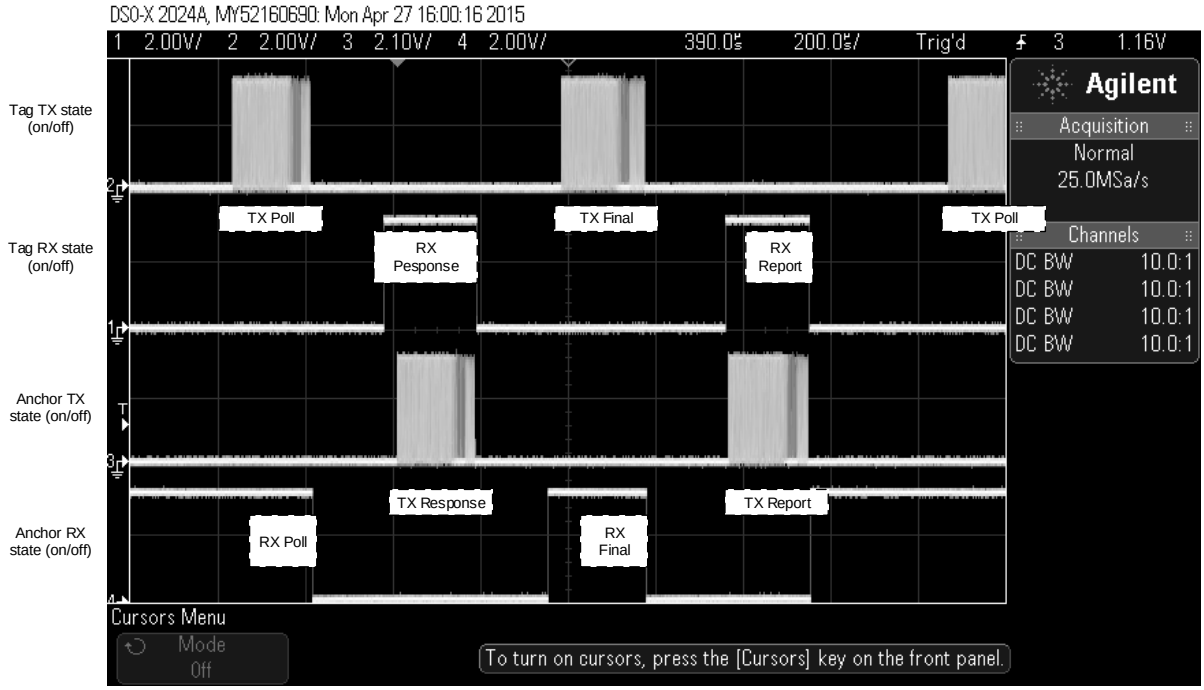

Figure 4: TX and RX states during two-way ranging

Figure 4 shows TX and RX states on GPIOs 5 and 6 during a basic two-way ranging (TWR) exchange (as described fully in DW1000 User Manual [2]) between two devices (a tag and an anchor) which uses three messages (`Poll`, `Response` & `Final`).

In Figure 4, channels 1 and 2 display the tag's RX and TX states, and channel 3 and 4 display the anchor's TX and RX states respectively. At the start of the TWR exchange the anchor is in the RX state awaiting messages. The tag transmits a `Poll` message, after which both the tag and the anchor are in IDLE. Then the tag turns its receiver on to receive the anchor's `Response` message after which the two devices return to the IDLE state. Finally the anchor turns its receiver on again to receive the tag's `Final` message which completes the TWR exchange.

4 USING STATUS (SYS_STATUS) AND STATES (SYS_STATE) REGISTERS

4.1 Introduction

The section of the DW1000 User Manual [2] dealing with the register set gives an overview and details of each of the DW1000 registers.

Here we concentrate on the system status (SYS_STATUS) and system states (SYS_STATE) registers and show how they can be used to let the developer know the internal state of the DW1000.

This information can be very useful in determining what the DW1000 is doing if it is not performing as expected.

4.2 The System Status (SYS_STATUS) register

This register contains status bits that indicate the occurrence of different system events or status changes. It is possible to enable particular events as interrupt sources, by employing the SYS_MASK register, so that the setting of the event status bit will generate an interrupt. This will assert the DW1000 hardware IRQ output line. The DW1000 User Manual [2] describes each of the status events in detail. Here we outline the functions of the most important events in the status register:

Table 1: Main status register events

System Event	Bit Number	Description
IRQS	0	Interrupt status (1 means there are pending events / interrupts, 0 means interrupt line shows no pending events).
AAT	3	Set to indicate the ACK has been sent.
TXFRS	7	If set this means that the frame has been sent.
RXPRD	8	Received preamble / preamble detected.
RXSFDD	9	Received SFD / SFD detected.
LDEDONE	10	LDE has finished and updated the RX timestamp.
RXPHD	11	Received PHY header.
RXPHE	12	Detected PHY header error.
RXDFR	13	End of frame. The data in RX buffer is ready for reading.
RXFCG	14	Received frame and good CRC.
RXFCE	15	Received frame and error in CRC.
RXFSL	16	Frame sync loss, frame error.
RXRFTO	17	RX frame timeout. Indicates that no frame was received within the time specified.
LDEERR	18	LDE error.
RXPTO	19	Preamble timeout. (This will happen after RX enable if no preamble detected within specified number symbols – see DRX_PRETOC in [2])
RXSFDTO	26	SFD timeout. (This will follow RXPRD if no SFD detected within specified number of symbols – see register DRX_SFDTOC in DRX_CONF in [2])
HPDWARN	26	“late” error when using delayed TX start / RX enable.
AFFREJ	29	Frame filtering rejection. (This will follow RXPHD if the decoded frame data does not match the configured filtering rules).
TXPUTE	34	TX power up error. This can happen if the delayed transmission

System Event	Bit Number	Description
		time is configured to be in the near future so that the TX blocks cannot power up in time for the transmission of the first symbol. Also see DW1000 User Manual [2].

4.3 The System States (SYS_STATE) register

This register contains information on the current DW1000 state.

The states are divided into 3 main types: TX, RX or IDLE and are shown in Table 2. Receiver states are shown in Table 3 while transmitter states are shown in Table 4 below.

Using the system status and states register the user can check what state the DW1000 is in, e.g. if a RX enable command has been issued, the states register will report 0x050500 (RX) and then once a frame has been received the register will report 0x010000 (IDLE). The SYS_STATE register bits are described below:

4.4 Overall system states

Table 2: Bits 23:16 PMSC_STATE Current PMSC State Machine value

Bit Value	State	Description
0x0	INIT	DW1000 is in INIT
0x1	IDLE	DW1000 is in IDLE
0x2	TX WAIT	DW1000 is waiting to start transmitting
0x3	RX WAIT	DW1000 is waiting to enter receive mode
0x4	TX	DW1000 is transmitting
0x5	RX	DW1000 is in receive mode

4.5 Receiver states

Table 3: Bits 15:8 RX_STATE Current Receive State Machine value

Bit Value	State	Description
0x00	IDLE	Receiver is in idle
0x01	START ANALOG	Start analog receiver blocks
0x04	RX READY	Receiver ready
0x05	PREAMBLE FIND	Receiver is waiting to detect preamble
0x06	PREAMBLE TO	Preamble timeout
0x07	SFD FOUND	SFD found
0x08	CONFIGURE PHR RX	Configure for PHR reception
0x09	PHR RX START	PHR reception started
0x0A	DATA RATE READY	Ready for data reception
0x0C	DATA RX SEQ	Data reception
0x0D	CONFIG DATA	Configure for data
0x0E	PHR NOT OK	PHR error
0x0F	LAST SYMBOL	Received last symbol
0x10	WAIT RSD DONE	Wait for Reed Solomon decoder to finish
0x11	RSD OK	Reed Solomon correct
0x12	RSD NOT OK	Reed Solomon error
0x13	RECONFIG 110	Reconfigure for 110 kbps data

Bit Value	State	Description
0x14	WAIT 110 PHR	Wait for 110 kbps PHR

4.6 Transmitter states

Table 4: Bits 7:0 TX_STATE Bits 7:4 - Reserved, 3:0 Current Transmit State Machine value

Bit Value	State	Description
0x0	IDLE	Transmitter is in idle
0x1	PREAMBLE	Transmitting preamble
0x2	SFD	Transmitting SFD
0x3	PHR	Transmitting PHR
0x4	SDE	Transmitting PHR parity SECDED bits
0x5	DATA	Transmitting data
0x6	RSP DATA	Transmitting Reed Solomon parity block

5 DW1000 TRANSMISSION

5.1 Introduction

After the DW1000 IC is powered up its registers will be configured to their default values (we are assuming here that the device is powered up from cold, i.e. has not been in DEEPSLEEP mode). These are described in the DW1000 User Manual [2], see section on *Default Configuration on Power Up*. In this chapter we'll assume that SPI communication with DW1000 has been verified and is working correctly.

The DW1000 IC is a transceiver, thus it can be commanded to either send a frame or enable its receiver to receive frames.

In active operation state, the DW1000 can either be in TX or RX mode. It cannot do both at the same time. If we need to change from TX to RX or vice versa then TRXOFF command must be used.

5.2 Basic Operation

To configure the DW1000 to transmit a frame please follow the **"**DW1000 Application Programming Interface with application examples" – namely ***ex_01a_simple_tx***. This example guides the user though the necessary register reads / writes to enable transmission.

To verify that a transmission has been successful, after initiating the transmission, the status register (SYS_STATUS) can be polled and checked to verify that TX events are set:

- TXFRB (frame begins)
- TXPRS (preamble sent)
- TXPHS (PHY header sent)
- TXFRS (frame sent).

These are described fully in the DW1000 User Manual [2].

If GPIO 5 is configured as EXTTXE and monitored on the oscilloscope the pin will go high when the transmission is active, as shown in Figure 3. The power / current profile can also be used as a guide to check which state the DW1000 is in (e.g. TX, IDLE) as shown in Figure 6.

To confirm successful transmission, set up an EVB1000 connected to a PC running DecaRanging software. Configure the EVB1000 to Listener mode and confirm that packets are successfully received. Note that the configuration parameters in the Listener must be the same as the transmitter (channel, preamble length, PRF etc.) for it to receive transmissions.

5.3 Supported transmission types

The DW1000 supports two different types of transmission: Immediate and Delayed.

5.3.1 Immediate TX

Immediate transmission means that the DW1000 will start sending a frame once the "start TX" command has been sent to it by the external controller. The DW1000 will power up any necessary TX blocks in the correct sequence and start sending the frame as soon as possible.

5.3.2 Delayed TX

The DW1000 can also use the delayed transmission option. This means that the transmission will happen in the future at a preconfigured system clock time (as written to the DX_TIME register). The operation of delayed TX is further described in the DW1000 User Manual [2] (section on Delayed Transmission).

NOTE on HPDWARN event status:

The HPDWARN event status flag is fully described in the DW1000 User Manual [2]. It basically warns of the "lateness" condition (i.e. microprocessor was too late in issuing the TX start command). If this happens, the `dwt_starttx` API function will return an error (see TX events section below where this is explained in more detail), and the DW1000 transmission will fail. The IC will be in IDLE state.

5.4 TX events

When DW1000 is instructed to send a frame, once the frame is sent the TXFRS will be set in the status register. Before the DW1000 is given the TX start command, the frame control and frame data registers should be configured. The API functions `dwt_writetxdata`, and `dwt_writetxfctrl` do this. The code below shows an example of sequence of function calls for immediate transmission.

```
dwt_writetxdata()
dwt_writetxfctrl()
dwt_starttx(DWT_START_TX_IMMEDIATE)
```

If the status register is read before the `dwt_starttx` function the TXFRS bit will be clear (assuming there were no previous transmissions, or if there were, the status register TXFRS bit has already been cleared), and then if the register is polled or read after the interrupt has triggered, the TXFRS bit will be set once the frame transmission has completed.

The timing of TXFRS will depend on the length of the frame being transmitted; e.g. if a 128 symbol preamble is used and a 20 byte payload is transmitted at 6.81 Mbps data rate then the entire frame will be transmitted in 0.2 ms. If the same 20 byte payload is transmitted at 110 kbps data rate and using a 1024 symbol preamble then the frame will take more than 2 ms to transmit.

Note: if the DW1000 transmits a frame and the TXFRS bit is not cleared, it will be automatically cleared on the start of the next transmission, however this will take about 6 µs from the time the *starttx* command is issued.

The code below shows an example of a sequence of function calls for delayed transmission.

```
//to send a frame at calculate delayed transmission time of DLYT
dwt_writetxdata()
dwt_writetxfctrl()
dwt_setdelayedtrxtime(DLYT)
late = dwt_starttx(DWT_START_TX_DELAYED)
```

If `late` is 0, then the DW1000 will send a frame at the time specified and set by `dwt_setdelayedtrxtime` function. If the status register is read before the `dwt_starttx` function the TXFRS bit will be clear, and then if the register is polled the TXFRS bit will be set once the frame transmission has completed.

If `late` is 1, then the DW1000 reported HPDWARN or TXPUTE event (inside the `dwt_starttx` function), and the transmission has aborted (SYS_CTRL_TRXOFF was issued). The DW1000 will be in IDLE after `dwt_starttx` call.

The diagram below (Figure 5) shows the minimum time needed to send a command to DW1000 in advance of the delayed transmission time so that the delayed TX will be successful. The time at which the TXSTRT command gets issued (Ts) has to take into account the Ton which depends on the frame length (preamble and SFD). The delayed time of sending (as programmed in register DX_TIME, and will be given by RMARKER) has to be > Ts + Ton. If RMARKER < Ts + Ton then the `dwt_starttx` will return "late" error.

The *dwt_starttx* function can be modified to remove the HPDWARN check, then the DW1000 will send the frame at time programmed, even if it is more than half of the clock period into the future. Note: The TXPUTE event should always be checked, and it if is set, the transmission must be aborted

(SYS_CTRL_TRXOFF issued) otherwise the DW1000 may never issue a TXFRS event.

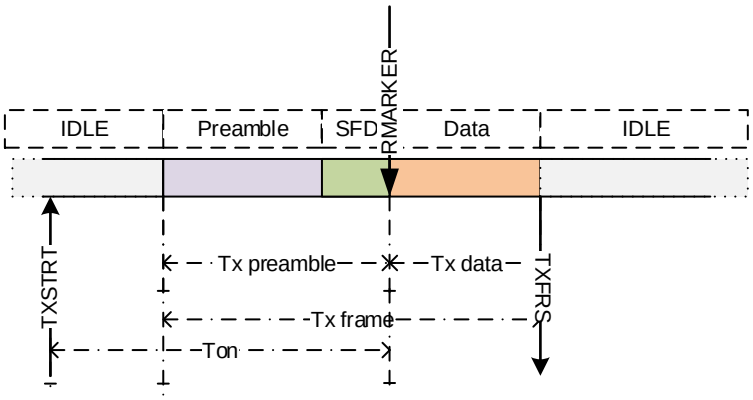

Figure 5: TX frame command timings (delayed TX operation)

5.5 TRX off

If the application needs to stop transmission of a frame, the TRXOFF command (in SYS_CTRL register) must be used. When this is issued the DW1000 returns to the IDLE state immediately. Any TX activity that is in progress at that time will be aborted. The TRXOFF bit will clear as soon as the DW1000 sees it and returns the IC to idle mode.

5.6 TWR current profile

As shown in the DW1000 Data Sheet [1], the DW1000 draws varying amounts of current depending on its operational state. As a debugging tool the current profile can be used to check DW1000 operation. Figure 6 shows a typical TWR current profile for a tag as it is transitioning between TX, RX, IDLE and SLEEP states. Initially the tag starts off in the IDLE state while it is getting ready to transmit the `Poll` message, then it turns on its receiver to await the `Response` message after which it will stay in IDLE until it sends the `Final` message and (can optionally) turn its receiver back on to receive the `Time of Flight Report` message as shown, before entering SLEEP state again.

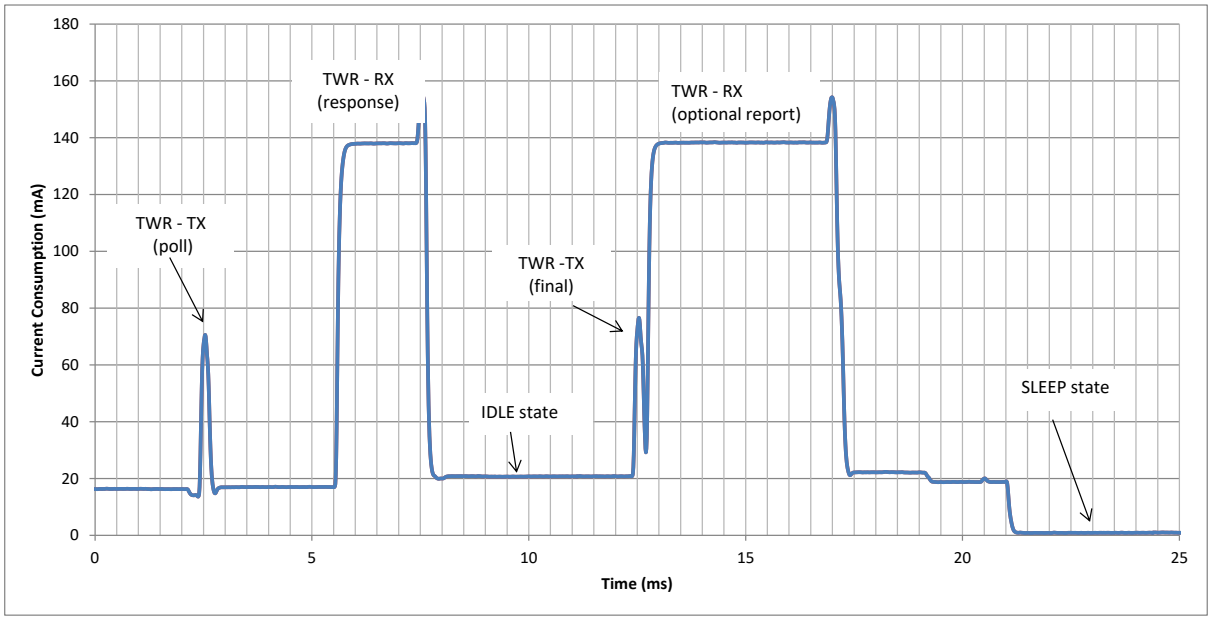

Figure 6: TWR power profile for a TWR tag using a DC-DC converter. Channel 5, 6.8 Mbps, 128 symbol preamble

Here it can be seen that the maximum current drawn is when the DW1000 is in the receive state, when demodulating data. The least amount is in sleep. The current profile can be measured using a DC power analyser for example the Agilent N6705B.

6 DW1000 RECEPTION

6.1 Introduction

The previous chapter described DW1000 transmission modes and their operation, this chapter deals with reception modes.

6.2 Basic Operation

To configure the DW1000 to receive a frame please follow the **"**DW1000 Application Programming Interface with application examples" – namely ***ex_02a_simple_rx***. This example guides the user though the necessary register reads / writes to enable reception. To confirm reception is operating correctly you could use a transmitting device such as an EVB1000 controlled by PC DecaRanging software.

After initiating reception, to verify that the reception is successful the status register (SYS_STATUS) can be polled and checked to verify that RX events are set:

- RXPRD (preamble detected)
- RXSFDD (SFD detected)
- RXPHD (PHY header detected)
- RXPHE (PHY header error)
- RXDFR (data frame ready)
- RXFCG (FCS good)
- RXFCE (FCS error) etc.

If GPIO 6 is configured as EXTRXE and monitored on an oscilloscope the pin will go high when the receiver is active, as shown in Figure 3. The current profile can also be used as a guide to check which state the DW1000 is in (e.g. RX, IDLE), as shown in Figure 6.

6.3 Supported reception types

The DW1000 supports different types of reception: Immediate, Delayed and Auto RX after TX.

6.3.1 Immediate RX

Immediate reception means that the DW1000 receiver will turn on once the start RX command has been sent to it. There is a short (16 µs) internal DW1000 delay before the receiver is able to receive anything.

6.3.2 Delayed RX

DW1000 can use delayed reception by configuring the DX_TIME register before writing delayed RX command, similar to the delayed TX operation.

NOTE on HPDWARN event status:

The HPDWARN event status flag is fully described in the DW1000 User Manual [2], it basically warns of the "lateness" condition (i.e. microprocessor was too late in issuing the RX start command). If this happens the *dwt_rxenable* API function will either turn on the receiver immediately, or put DW1000 into IDLE depending on the input parameter and also report an error (please see the API function description in [3]).

6.3.3 Auto RX after TX

The DW1000 can automatically enter receive mode once the transmission completes. This can be immediate or delayed depending on the configuration of ACK_RESP_T register. For more information on this see WAIT4RESP bit description of SYS_CTRL register in the DW1000 User Manual [2].

6.4 RX events

When DW1000 is instructed to enable the receiver, it will go into the RX state**. It will stay in the RX state until it times out, receives an error or receives a good frame.** When the DW1000 exits the receive state it goes back to the IDLE state.

After a receive enable command is given, the status register can be polled or the interrupt line can be monitored until one of the receive events is triggered. If timeouts are used, they need to be configured before the receiver is enabled. The error events and timeouts (apart from RXRFTO) will only trigger if the auto RX re-enable is not used. For more information on auto RX re-enable see RXAUTR bit description of SYS_CFG register in the DW1000 User Manual [2].

6.5 TRX off

If the application needs to stop reception of a frame, the TRXOFF command (in SYS_CTRL register) must be used. When this is issued the DW1000 returns to the IDLE state immediately. Any RX activity that is in progress at that time will be aborted. The TRXOFF bit will clear as soon as the DW1000 sees it and returns the IC to idle mode.

6.6 TWR power profile

As shown in the previous chapter, section 5.6 the DW1000 consumes varying amounts of current depending on its operational state. As a debugging tool the current profile can be used to check DW1000 operation.

7 DW1000 SLEEP & DEEPSLEEP STATES

7.1 Introduction

The operational states of the DW1000 are defined in the DW1000 User Manual [2]. They are OFF, WAKEUP, INIT, IDLE, SLEEP, DEEPSLEEP, TX, RX and SNOOZE.

This section briefly describes the SLEEP, DEEPSLEEP & WAKEUP states and the configuration required to transition from SLEEP or DEEPSLEEP to WAKEUP.

7.2 SLEEP

While in this state the DW1000 internal low power oscillator is running and is used to clock the sleep counter whose expiry is programmed to “wake up” the DW1000 and progress into the WAKEUP state. In this state, the DW1000 consumes less than 1 µA from the external power supplies.

7.3 DEEPSLEEP

While in this state all of the DW1000 internal circuitry is powered down with the exception of the always-on memory which can be used to hold the device configuration for restoration on wakeup. The DW1000 will remain in this state until the occurrence of a wakeup event.

7.4 WAKEUP

Upon entering the WAKEUP state the crystal oscillator and the band-gap voltage reference are enabled. After approximately 4 ms the digital LDO will be enabled and the RSTn (output) will de-assert allowing the DW1000 to enter the INIT state. Once in the INIT state the AON configuration will be restored automatically. This can take up to 40 µs (for downloading AON array and for the on-chip PLL to lock).

During this time SPI access to the device should not be attempted as the values read back may be invalid or corrupted.

7.5 Configuring WAKEUP and DEEPSLEEP

While in the DEEPSLEEP state the DW1000 consumes very little current. If the DW1000 needs to perform only intermittent operations then use of the DEEPSLEEP state is recommended.

Before entering the SLEEP or DEEPSLEEP states the wake up options must be configured using the `dwt_configuresleep` function call. These options specify: -

- What event will be used to wake up the device
- What the device will do once it has woken up

The code below shows an example sequence of function calls for configuration of WAKEUP and DEEPSLEEP. Table 6 shows the configuration options as used by the `dwt_configuresleep` function.

The **`wake`** parameter in the `dwt_configuresleep` function call defines what event will be used to wake up the DW1000. There are three options to wake up the device (see Table 5): -

- Expiry of the SLEEP counter (only applies to SLEEP mode not to DEEPSLEEP);
- The WAKEUP pin being brought high for the required time; or
- Activity on the SPICSn input

Table 5: Bitmask values for dwt_configuresleep() *wake* bit mask

Event	Bit mask	Description
DWT_WAKE_SLPCNT	0x8	Wake up after sleep count expires. By default this configuration is set enabling the sleep counter as a wake-up signal. Setting this configuration bit to 0 will mean that the sleep counter cannot awaken the DW1000 from SLEEP.
DWT_WAKE_CS	0x4	Wakeup on the SPICSn chip select line.
DWT_WAKE_WK	0x2	Wake up on WAKEUP line.
DWT_SLP_EN	0x1	This is the sleep enable configuration bit. This needs to be set to enable DW1000 SLEEP/DEEPSLEEP functionality.

The **mode** parameter in the *dwt_configuresleep* function call defines what the DW1000 will do on wake-up.

Table 6: Bitmask values for dwt_configuresleep() *mode* bit mask

Event	Bit mask	Description
DWT_LOADLDO	0x1000	Load LDO tuned value after wake up.
DWT_LOADUCODE	0x0800	Load micro-code for the LDE engine.
DWT_PRESRV_SLEEP	0x0100	Preserves sleep. When this is set these sleep controls are not cleared upon wakeup, so that the DW1000 can return to sleep, for example after a failed reception.
DWT_LOADOPSET	0x0080	On wakeup load the receiver operating parameter When the bit is 0 the receiver operating parameter set reverts to its power-on-reset value (the default operating parameter set) when the DW1000 wakes from SLEEP or DEEPSLEEP.
DWT_CONFIG	0x0040	Restore saved configurations.
DWT_LOADEUI	0x0008	On Wake-up load the EUI value from OTP memory into register 0x01. The 64-bit EUI value will be stored in register 0x01 when the DW1000 wakes from DEEPSLEEP or SLEEP states.
DWT_GOTORX	0x0002	On Wake-up turn on the receiver. With this bit it is possible to make the IC transition into RX automatically as part of IC wake up.
DWT_TANDV	0x0001	On Wake-up run the (temperature and voltage) ADC. Setting this bit will cause the automatic initiation of temperature and input battery voltage measurements when the DW1000 wakes from DEEPSLEEP or SLEEP states. The sampled temperature value may be accessed using the *dwt_readtempvbat*

The sequence of events is as follows: -

Firstly, configure the WAKEUP option (in this case wake on WAKEUP pin) and select which mode to go into on wakeup:

```
mode = DWT_LOADUCODE|DWT_PRESRV_SLEEP|DWT_CONFIG|DWT_TANDV
wake = DWT_WAKE_WK|DWT_SLP_EN
dwt_configuresleep(mode,wake)
```

Then, when ready to enter DEEPSLEEP, call `dwt_entersleep` (this uploads the current register configuration to the AON array and puts the DW1000 to sleep, because DWT_SLP_EN has been selected as part of `wake` parameter above).

NOTE: now the DW1000 is in the DEEPSLEEP state – no SPI communications should be attempted

When the DW1000 is in the DEEPSLEEP state, SPI communications with it are not possible. If a register read is attempted, 0xFFFFFFFF will be returned. In order to communicate with the DW1000 it must be first put into the IDLE state by waking it up. The desired wake up option as described above needs to be configured before the device is put to sleep.

As the device wakes up the RSTn pin will go high after which the device will switch from INIT to IDLE mode.

If the device is in DEEPSLEEP and it needs to be reset, then it must first be woken up and put into IDLE mode before the reset is attempted.

Here the reader is invited to review the **"**DW1000 Application Programming Interface with application examples" – namely ***ex_01b_tx_sleep*** and ***ex_01c_tx_sleep_auto.***

8 REFERENCES

Reference is made to the following documents in the course of this document: -

Table 7: Table of References

Ref	Author	Version	Title
[1]	Decawave	Current	DW1000 Data Sheet
[2]	Decawave	Current	DW1000 User Manual
[3]	Decawave	Current	DW1000 Software API Guide

9 DOCUMENT HISTORY

Table 8: Document History

Revision	Date	Description
1.0	March 31st, 2016	Initial release
1.1	June 30th, 2016	Scheduled update
1.2	December 30th, 2016	Scheduled update

10 MAJOR CHANGES

v1.0

Page	Change Description
All	Initial release

v1.1

Page	Change Description
All	Typographical corrections
Cover	Update of revision number to 1.1
12	Corrections to section 4.3 / heading errors
19	Correction of “Error reference not found”
22	Addition of revision 1.1 to table 8
22	Addition of this section 10 and the two tables therein

v1.2

Page	Change Description
Cover	Update of revision number to 1.2
4	Update Coocox link
8	Change API function call reference due to change in API function name new function is `dwt_setlnapamode(1,1)` to configure GPIOs 5 and 6 as TX and RX states.

v1.3

Page	Change Description
All	New logo and template.
23	New section for “Further Information”
First page	New revision 1.3 and addition to Revision table for Document History.

11 FURTHER INFORMATION

Decawave develops semiconductors solutions, software, modules, reference designs - that enable real-time, ultra-accurate, ultra-reliable local area micro-location services. Decawave’s technology enables an entirely new class of easy to implement, highly secure, intelligent location functionality and services for IoT and smart consumer products and applications.

For further information on this or any other Decawave product, please refer to our website www.decawave.com.