FRONTGRADE

APPLICATION NOTE

UT32M0R500

Determining External Clock Accuracy for Asynchronous Peripherals

5/26/2022

Version #: 1.0.0

Table 1: Cross Reference of Applicable Products

Product Name	Manufacturer Part Number	Smd #	Device Type
Arm Cortex M0+	UT32M0R500	5962-17212	01, 02

1.0 Introduction

The UT32M0R500 UART and CAN peripherals do not include a clock signal in their data transmission, so they are considered asynchronous communication protocols. If the clock source provided to these peripherals drifts too far (increases or decreases in frequency), data will become unintelligible to other devices on the bus. Due to the accuracy requirements of these protocols, some designs may need to use a high accuracy external oscillator instead of the UT32M0R500's internal oscillator.

2.0 Calculating CAN Clock Drift Tolerance

The UT32M0R500 uses four different bit fields to determine the baudrate of the CAN peripheral. These are the Baud Rate Pre-scaler (BRP), Synchronization Jump Width (SJW), Time Segment 1 (TSEG1), and Time Segment 2 (TSEG2). These values are also used to calculate the amount of oscillator drift (df) around a nominal frequency the CAN node can tolerate.

$$(1 - df) * fnom \leq fosc \leq (1 + df) * fnom$$

Two different conditions define the tolerable oscillator drift of a system. The first (I) equation quantifies the time for a node on the bus to resynchronize after receiving and echoing an error flag. When an active error occurs on the CAN bus, a node transmits six dominant bits in a row. Then, each node that sees the error flag responds with their own error flag. After these twelve bits, the next bit (bit thirteen) is the point in time where a node would need to resynchronize. The second (II) equation is the ability for a node to resynchronize after bit stuffing. CAN Bit stuffing ensures that for any given order of bits, the maximum number of dominant or recessive bits in a row is five (six dominant bits is an error message). Therefore, a system would need to resynchronize after a total of ten bits (5 dominant followed by five recessive, followed by a dominant bit). See Reference [4] for a more verbose explanation.

I: $$df \leq \frac{\min \leq (t_{TSEG1}, t_{TSEG2})}{2 * (13 * t_{BIT} - t_{TSEG2})}$$

II: $$df \leq \frac{t_{SJW}}{2 * 10 * t_{BIT}}$$

The value of df must be true for both equations.

2.1 Example Calculation

UT32M0R500 Settings:

- f_{NOM} = 50MHz
- t_{NOM} = 20ns
- BRP = -0x4
- TSEG1 = 0x1
- TSEG2 = 0x1
- SJW = 0x1

Calculating Components of the maximum drift tolerance equations:

- Time quanta (tq) = 2 * t_{OSC} * (BRP+1) = 2 * 20ns * (5) = 200ns
- t_{SJW} = (SJW+1) * tq = 2 * 200ns = 400ns
- t_{PROP_SEG} = 1 * tq = 200ns
- t_{TSEG1} = (TSEG1 + 1) * tq = 2*200ns = 400ns
- t_{TSEG2} = (TSEG2 + 1) * tq = 2*200ns = 400ns
- t_{BIT} = t_{PROP_SEG} + t_{TSEG1} + t_{TSEG2} = 1000ns
- BAUD = 1Mbps

Clock Drift Equations:

$$\text{I:}\quad df \leq \frac{\min(400ns, 400ns)}{2 * (13 * 1000ns - 400ns)} = \frac{400ns}{2 * (13000ns - 400ns)} = \frac{400ns}{25200ns} = 0.01587$$

$$\text{II:}\quad df \leq \frac{2}{2 * 10 * 1000ns} = 0.02$$

In order for equation I and II to be true, df must be ≤ 0.0158, or 1.587%. Applying that value to the original equation of fnom and fosc:

$(1 - 0.01587) * 50MHz \leq fosc \leq (1+0.01587) * 50MHz$

$49.2063MHz \leq fosc \leq 50.7936MHz$

Appendix B provides a table of calculations for the UT32M0R500 with a 50MHz and 48MHz nominal oscillator, using select values for 1Mbps, 500kbps, 250kbps, and 125kbps.

3.0 Calculating UART Clock Drift Tolerance

The UT32M0R500 UART peripheral uses a 12-bit scaler to determine the baud rate. The UT32M0R500 oversamples each baud eight times. The following formula looks at the minimum and maximum frequency, as effected by the Baud Tolerance and Baud Offset Error. fnom is the nominal clock frequency, and fosc is the operating clock frequency.

$$(1 - (-Baud_{Tolerance} - Baud_{Offset\ Error})) * fnom \leq fosc \leq (1 + (Baud_{Tolerance} - Baud_{Offset\ Error})) * fnom$$

The Baud Tolerance is the amount of oscillator drift in either direction the UT32M0R500 can handle. The Baud Offset Error is the error between the target Baud and actual Baud of the UT32M0R500. When calculating the correct Scaler value to use for a given Baud rate, use the following:

$$Scaler_{Calculated} = \frac{fnom}{(Baud_{Target} * 8)} - 1$$

Because the scaler bits of the UT32M0R500 can only represent integer values, the calculated scaler needs to be rounded to determine the actual Baud. C rounds decimal values into integers by truncating the integer, so the programmed scaler value is equal to the calculated scaler value rounded down. Users could also round up the programmed scaler but would need to implement this manually in their project.

$$Scaler_{Programmed} = [Scaler_{Calculated}]$$

$$Baud_{Actual} = \frac{fnom}{(Scaler_{Programmed}+1)*8}$$

The Baud Offset Error is equal to the percentage difference between the target Baud and actual Baud.

$$Baud_{OffsetError} = \frac{Baud_{Actual} - Baud_{Target}}{Baud_{Target}}$$

The UT32M0R500 UART messages always have 1 start bit, 8 data bits, and 1 stop bit. Optionally, they can have 1 parity bit. In order for a UART receiver to successfully receive a message, the last sample clock needs to occur during the stop bit. Each bit is supposed to be sampled in the middle of the hold time. For a no parity message, a receiver would attempt to sample the stop bit after 76 clock periods ((1 start + 8 data + 0.5 stop bits) * 8 clocks/bit). For a parity message, this would occur after 84 clock periods ((1 start + 8 data + 1 parity + 0.5 stop bits) * 8 clocks/bit).

Assuming an average UART system, where the hold time of a bit is 75% of the total bit time (allowing 25% for transition on either end of the bit), there is a ±37.5% tolerance around the middle of a bit, equivalent to ±3 clock periods. However, if the falling edge of a Start bit occurs just after a fosc rising edge, that tolerance is reduced to ±2 clock periods. This will be referred to as the clock period error. The Baud Tolerance is equal to the number of clock periods of tolerance divided by the total number of clock periods in a message.

$$\text{Clock Period Error} = \left(\frac{\text{Hold Time Window}}{2} * 8 \text{ Clocks per bit}\right) - 1$$

$$Baud_{Tolerance} = \frac{\text{Clock Period Error}}{\text{PCLKs from Start Edge to Stop Sample}}$$

3.1 Example Calculation

UT32M0R500 Settings:

- fNOM = 50MHz BaudTarget = 19200
- Hold Time Window = 75%
- Parity bits = 0

Start by finding the Baud Offset Error. To do so, calculate the value programmed into the UART Scaler register for the targeted baud.

$$Baud_{OffsettError} = \frac{Baud_{Actual} - Baud_{Target}}{Baud_{Target}}$$

$$Scaler_{Calculated} = \frac{50MHz}{(19200*8)-1} = 325.5229$$

$$Baud_{Actual} = \frac{50MHz}{(325 + 1) * 8} = 19171.779$$

$$Baud_{OffsetError} = \frac{19171.779-19200}{19200} = -0.00146 \ \textit{or} \ -0.146\%$$

Next, find the Baud Tolerance:

$$Baud\ _{Tolerance} = \frac{\text{Clock Period Error}}{\text{PCLKs from Start Edge to Stop Sample}}$$

$$ClockPeriodError = \left(\frac{0.75}{2} * 8\right) - 1 = \pm 2$$

$$Baud_{Tolerance} = \frac{\pm 2}{76} = \pm\ 0.02631 \ \textit{or} \ \pm\ 2.631\%$$

Finally, find the minimum and maximum drift values for the oscillator:

- $(1 - (-0.02631 - (-0.00146))) * 50MHz \leq fosc \leq (1 + (0.02631 - (-0.00146))) * 50MHz$
- $(1 - 0.0248) * 50MHz \leq fosc \leq (1 + 0.0277) * 50MHz$
- $48.7577\ MHz \leq fosc \leq 51.3892\ MHz$

Because the difference between programmed and target baud rates, there is more tolerance for positive oscillator drift than there is for negative oscillator drift.

4.0 External vs Internal Oscillator

Based upon the calculations in Appendix B and C, designs that plan on using the UT32M0R500's CAN peripheral should use an external clock with a ±1.5% tolerance. If the design isn't using CAN, but will use UART, the external clock tolerance can be widened to ~±2.0% drift for most BAUD rates. The internal oscillator of the UT32M0R500 has a specified accuracy of ±4%. While at room temperature the UT32M0R500 is more or less stable, but as aging, temperature, or radiation is applied the frequency will change. Therefore, the internal Oscillator isn't suitable for asynchronous communication such as CAN or UART in extreme conditions.

For systems that are using UART, software could theoretically fine-tune the scaler value to achieve incremental adjustments during user code operation. For systems using CAN, the adjustments that could be made have too different of a Baud rate compared to their drift tolerance to allow for continuous coverage of a linear baud drift.

5.0 Conclusion

Systems using asynchronous peripherals such as UART and CAN should use a high accuracy external oscillator as the core clock of the UT32M0R500 to ensure message retention between devices. For information on how to use an external oscillator with the UT32MR0500, users can reference the UT32M0R500 Datasheet, the UT32M0R500 Evaluation Board Users Guide, and UT32M0R500 Functional Manual, all found on the Frontgrade website.

Appendix A: References

CAN Oscillator Timing Reliability

1. http://www.oertel-halle.de/files/cia99paper.pdf
2. https://www.can-cia.org/fileadmin/resources/documents/proceedings/2013_mutter.pdf
3. https://www.nxp.com/docs/en/application-note/AN1798.pdf
4. https://www.st.com/resource/en/application_note/an2352-pll-jitter-effects-on-ccan-modules-of-thest10f27x-stmicroelectronics.pdf (this has an error in the section 1.6.1 calculations)

UART Oscillator Timing Reliability

5. https://www.maximintegrated.com/en/design/technical-documents/tutorials/2/2141.html
6. https://erg.abdn.ac.uk/users/gorry/eg3576/UART.html
7. http://www.robotroom.com/Asynchronous-Serial-Communication-2.html
8. https://www.allaboutcircuits.com/technical-articles/the-uart-baud-rate-clock-how-accurate-does-it-need-tobe/

UT32M0R500 Relevant Documentation

9. https://caes.com/sites/default/files/documents/Datasheet-UT32M0R500.pdf
10. https://caes.com/sites/default/files/documents/Functional-Manual-UT32M0R500.pdf
11. https://caes.com/sites/default/files/documents/App-Note-UT32M0R500-EVB-Users-Guide.pdf

Appendix B: CAN Timing Calculations for Common Baud Rates

The below calculations are not a complete list, and focus on configurations that have a df ≥ 1.5% for 1000, 500, 250, and 125 kbps.

UT32M0R500 Configuration (BRP/TSEG1/TSEG2/SWJ)	fnom (MHz)	Bit Rate (Kbps)	df1	df2	df = min(df1,df2)	fosc * (1-df)	fosc * (1+df)
4/1/1/1	50	1000	0.01587	0.02	0.01587	49.2063	50.7936
4/3/4/3	50	500	0.016	0.02	0.01600	49.2000	50.8000
4/4/3/3	50	500	0.01587	0.02	0.01587	49.2063	50.7936
9/1/1/1	50	500	0.01587	0.02	0.01587	49.2063	50.7936
9/3/4/3	50	250	0.016	0.02	0.01600	49.2000	50.8000
9/4/3/3	50	250	0.01587	0.02	0.01587	49.2063	50.7936
19/1/1/1	50	250	0.01587	0.02	0.01587	49.2063	50.7936
19/3/4/3	50	125	0.016	0.02	0.01600	49.2000	50.8000
19/4/3/3	50	125	0.01587	0.02	0.01587	49.2063	50.7936
39/1/1/1	50	125	0.01587	0.02	0.01587	49.2063	50.7936
24/2/3/2	50	125	0.015	0.01875	0.01500	49.2500	50.7500
1/4/5/3	48	1000	0.01666	0.01666	0.01666	47.2000	48.8000
1/5/4/3	48	1000	0.01655	0.01666	0.01655	47.2052	48.7947
4/1/1/1	48	1000	0.01587	0.02	0.01587	47.2380	48.7619
2/2/3/2	48	1000	0.015	0.01875	0.01500	47.2800	48.7200
3/4/5/3	48	500	0.01666	0.01666	0.01666	47.2000	48.8000
3/5/4/3	48	500	0.01655	0.01666	0.01655	47.2052	48.7947
5/2/3/2	48	500	0.015	0.01875	0.01500	47.2800	48.7200
7/4/5/3	48	250	0.01666	0.01666	0.01666	47.2000	48.8000
7/5/4/3	48	250	0.01655	0.01666	0.01655	47.2052	48.7947
11/2/3/2	48	250	0.015	0.01875	0.01500	47.2800	48.7200
15/4/5/3	48	125	0.01666	0.01666	0.01666	47.2000	48.8000
15/5/4/3	48	125	0.01655	0.01666	0.01655	47.2052	48.7947
23/2/3/2	48	125	0.015	0.01875	0.01500	47.2800	48.7200

Appendix C: UART Timing Calculations for Common Baud Rates

Target Data Rate (bps)	fnom (MHz)	UT32M0R500 Configuration (Scaler, Parity bit)	Hold Time Window	Calculated Data Rate (bps)	Baud Offset Error	Baud Tolerance	Min fosc (MHz)	Max fosc (MHz)
9600	50	651, no parity	75%	9585.889	-0.0014	0.0263	48.7577	51.3892
9600	50	651, parity	75%	9585.889	-0.0014	0.0238	48.8830	51.2639
19200	50	325, no parity	75%	19171.779	-0.0014	0.0263	48.7577	51.3892
19200	50	325, parity	75%	19171.779	-0.0014	0.0238	48.8830	51.2639
38400	50	162, no parity	75%	38343.558	-0.0014	0.0263	48.7577	51.3892
38400	50	162, parity	75%	38343.558	-0.0014	0.0238	48.8830	51.2639
57600	50	108, no parity	75%	57339.449	-0.0045	0.0263	48.9103	51.5419
57600	50	108, parity	75%	57339.449	-0.0045	0.0238	49.0356	51.4166
115200	50	54, no parity	75%	113636.363	-0.0135	0.0263	49.3628	51.9944
115200	50	54, parity	75%	113636.363	-0.0135	0.0238	49.4881	50.0933
9600	48	625, no parity	75%	9584.664	-0.0015	0.0263	46.8135	49.3398
9600	48	625, parity	75%	9584.664	-0.0015	0.0238	46.9338	49.2195
19200	48	312, no parity	75%	19169.329	-0.0015	0.0263	46.8135	49.3398
19200	48	312, parity	75%	19169.329	-0.0015	0.0238	46.9338	49.2195
38400	48	156, no parity	75%	38216.560	-0.0047	0.0263	46.9661	49.4924
38400	48	156, parity	75%	38216.560	-0.0047	0.0238	47.0864	49.3721
57600	48	104, no parity	75%	57142.857	-0.0079	0.0263	47.1177	49.6441
57600	48	104, parity	75%	57142.857	-0.0079	0.0238	47.2380	49.5238
115200	48	52, no parity	75%	113207.547	-0.0172	0.0263	47.5670	50.0933
115200	48	52, parity	75%	113207.547	-0.0172	0.0238	47.6873	49.9730

Revision History

Date	Revision #	Author	Change Description	Page #
05/26/2022	1.0.0	OW	Initial Release	
10/30/2024	1.0.1	JA	Clock Drift Equations corrections	

Datasheet Definitions

	Definition
Advanced Datasheet	Frontgrade reserves the right to make changes to any products and services described herein at any time without notice. The product is still in the development stage and the **datasheet is subject to change**. Specifications can be **TBD** and the part package and pinout are **not final.**
Preliminary Datasheet	Frontgrade reserves the right to make changes to any products and services described herein at any time without notice. The product is in the characterization stage and prototypes are available.
Datasheet	Product is in production and any changes to the product and services described herein will follow a formal customer notification process for form, fit or function changes.