Keywords: UART, RS232, RS485, SPI, I2C, half duplex, HDX, full duplex, FDX, WLP, wafer level package, FIFO, CTS, RTS, clear to send, ready to send, baud

APPLICATION NOTE 5306

Programming Baud Rates of the MAX3108 UART

By: Micheal Scherrenburg
Mar 30, 2012

Abstract: The MAX3108 is a complete high-performance universal asynchronous receiver-transmitter (UART) in a tiny 2.1mm × 2.1mm wafer level package (WLP). It provides many high-end features in hardware, thus reducing firmware complexity. Instead of fast response and intensive firmware management, most of the MAX3108's advanced features need only initial firmware setup and little or no firmware management. This application note, one in a series explaining the MAX3108 features in detail, explains how to set target baud rates.

Introduction

The MAX3108, a high-performance universal asynchronous receiver-transmitter (UART) in a wafer-level package (WLP), is packed with many advanced features in hardware, from individual 128 word transmit and receive FIFOs to extensive hardware-mediated flow control. Most of the MAX3108's functionality relies on the programming of a clock. So, one of the first programming tasks is to set an appropriate baud rate clock.

This application note explains how to program the five registers (and 1 bit) to set a desired baud rate. A spreadsheet is provided that greatly simplifies the task of UART baud rate programming. This document presumes that the reader is already familiar with the application note 5304, "Interfacing to the MAX3108 UART."

Baud Rate Generation

This section describes how the MAX3108 creates a baud rate frequency from a crystal or oscillator input. This should be done first when programming the UART. Except for the SPI/I²C interface and basic GPIO functionality, nothing else in the MAX3108 functions until a clock rate is programmed. The device comes out of reset with clocks disabled if using a crystal as a frequency source.

The MAX3108 processes a clock source into an output baud rate in the following way:

1. If the clock source is a crystal, the crystal oscillator block converts this source to an internal logic level clock. The MAX3108 accepts crystals that resonate between 1MHz and 4MHz. To enable the crystal oscillator, set the CrystalEN bit to 1. Alternatively, an external logic level clock source may be provided at the XIN pin of the MAX3108. The MAX3108 accepts a clock frequency from 500kHz to 35MHz. Because the CrystalEn bit defaults to 0 at power-up or after reset, it will use an external clock unless specifically programmed otherwise.
2. The clock from step 1 feeds the input of a predivider. It divides the input frequency by a number between 1 and 63. Bits 5 through 0 of the PLLConfig register define the divisor.
3. The output of the predivider feeds a phase-locked loop (PLL), which multiplies the frequency of the predivider by

1, 6, 48, 96, or 144. To multiply by 1, set bit 2, the PLLEn bit of the CLKSource register, to 0. To set any other multiplier, set the PLLEn bit to 1, and also set bits 7 and 6 of the PLLConfig register as illustrated in Table 4 of the MAX3108 data sheet, which also details the frequency limits of the PLL. Note that the PLL cannot be used at all if the supply Voltage (V_{CC}) is less than 2.35V. The PLL is useful when generating high baud rates from low frequency clock sources.

4. The output of the PLL feeds a baud rate generator, whose output frequency is divided. This circuit divides its input frequency by an integer part (from 1 through 65,535) and a fractional part (from 0/16 through 15/16). This is programmed through the DIVMSB register, the DIVLSB register, and bits 3 through 0 of the BRGConfig register. The output of the baud rate generator is the internal baud rate clock. The actual baud rate would be the frequency output of the baud rate generator divided by 16. One can divide by 8 or by 4 instead, but dividing by 16 should be used if at all possible. Dividing by 8 or by 4 makes some features unusable, and reduces the frequency error tolerance of the MAX3108 UART receiver.

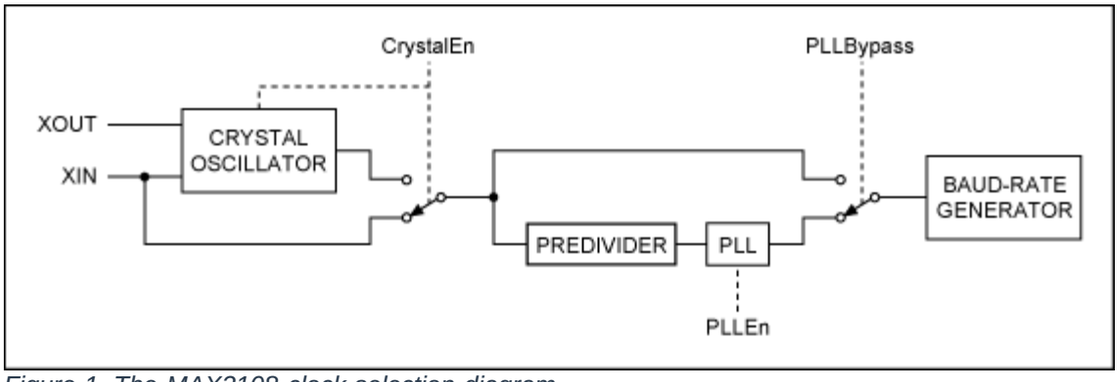

Figure 1. The MAX3108 clock selection diagram.

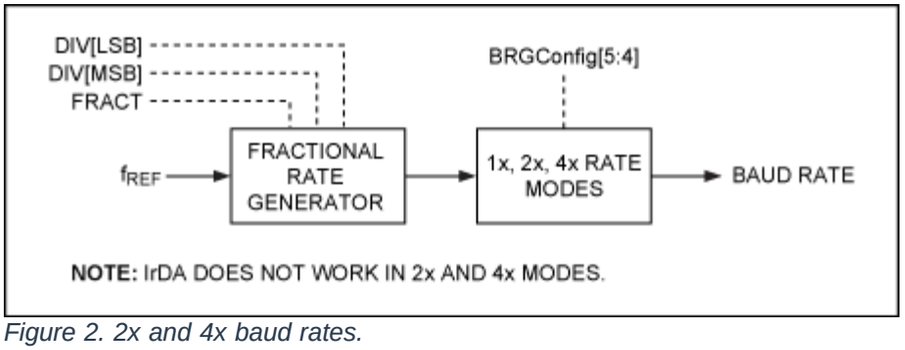

Figure 2. 2x and 4x baud rates.

Table 1. Differences between 1x, 2x, and 4x Rate Modes

Rate Mode Divider	BRGConfig[5]	BRGConfig[4]	IrDA Capable?	Line Noise Detect Capable?	Receive Frequency Error Tolerance
16	0	0	Yes	Yes	Best
8	0	1	No	Yes	Very good
4	1	0	No	No	Good

The crystal oscillator operates with common UART crystals, such as 1.8432MHz, and the external clock input works with a range of common clock frequencies found in embedded systems. This is a lot of information to digest just to

generate a baud rate clock. Also, for most baud rates, the same baud rate can be generated in many ways.

Baud Rate Assistance Spreadsheet

For this reason, the "MAX3108 Freq Select" spreadsheet is available for download. Based on a given input frequency and desired baud rate, this tool makes it easy to determine what to program into the MAX3108's five clock configuration registers. The spreadsheet calculates a few of the best ways to generate the target baud rate from the given input frequency.

In many cases, the target baud rate can be generated exactly. Sometimes however, the best possible configuration results in a small error in the generated baud rate. This is why the spreadsheet provides more than one possible configuration. Each possibility yields the closest frequency match, but with different assumptions.

Most of the time, there will be one configuration that is obviously superior. After describing the spreadsheet, we will work through some examples to show how easy it is to determine the best correct configuration to program into the MAX3108.

Spreadsheet Description

The spreadsheet needs only four inputs from the user, all near the top of the **Synthesize** tab.

	A	B	C	D
1	1.21			
2		**XIN Frequency:**	1843200	
3		**Desired BAUD Rate:**	230400	
4				
5		**Crystal (Y/N)?**	y	
6		**CLK on RTS PIN (Y/N)?**	n	
7				

Enter the four parameters as follows:

1. In cell C2, enter the crystal frequency or the external oscillator frequency, in Hertz.
2. In cell C3, enter the desired baud rate, in bits per second.
3. In cell C5, enter **y** if the frequency source is a crystal (that uses the MAX3108 crystal oscillator), or enter **n** if the frequency source is an external clock supplied to the MAX3108 XIN pin.
4. In cell C6, enter **n**. This pertains to a MAX3108 feature that will be described later.

The results appear starting in row 8.

	A	B	C	
7				
8		**XIN Test (XTAL):**	TRUE	
9		**XIN Test (Ext OSC):**	TRUE	
10				
11		Can it be done?	TRUE	

If cell C8 reports **FALSE**, then you have chosen a crystal as the frequency source, and it is out of the range of valid frequencies for the MAX3108 crystal oscillator circuit. Valid crystal oscillator frequencies range between 1MHz and 4MHz.

If cell C9 reports **FALSE**, then you have chosen an external clock source, and the frequency you have specified is outside the range of valid clock frequencies for the XIN pin. Valid external oscillator frequencies range between 0.5MHz and 35MHz.

Cell C11 reports if it is at all possible to reasonably generate the desired baud rate. Cell C11 can report **FALSE** because cells C8 or C9 report **FALSE**. Or because no configuration exists that can reasonably achieve the target baud rate. For example, requesting a baud rate above the MAX3108's maximum of 24MBd causes cell C11 to report **FALSE** regardless of the values of the other user entries.

Further results appear starting in row 13.

Columns D, E, and F calculate the configuration with the lowest frequency error, each under a different assumption. Column D assumes that the rate mode is fixed at 16, column E assumes that the rate mode is fixed at 8, and column F assumes that the rate mode is fixed at 4. As a general rule, the rate mode should be kept at 16, unless there is a compelling reason to choose otherwise, a topic that we will return to soon.

Row 14 is a flag, which if **FALSE**, indicates that the corresponding rate mode cannot be used in any configuration that produces a result near the desired baud rate. In this case, the other entries in this column will be blank.

Rows 15 through 17 and row 20 describe the configuration in a way that can be understood via Figures 1 and 2. Row 15 describes the predivider value chosen. Row 20 indicates which PLL multiplier was chosen. Additionally, rows 16 and 17 describe the integer and fractional part of the fractional rate generator. Of more direct interest, row 17, shows the actual baud rate generated, and row 18, contains the frequency error, in percent.

Rows 21 through 25 represent the values to program into the five MAX3108 clock configuration registers, in hex. When a configuration is selected by the user, these five hex values are copied into a program, to be described later, to load these values into the MAX3108.

A First Example

Let's begin with a simple example, generating 19.2kBd from a 1.8432MHz crystal. Enter the following parameters:

	A	B	C
1	1.21		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	19200
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n

This is the result:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		1	1	1
16		Best BRG INT:		6	12	24
17		Best BRG Frac:		0	0	0
18		Best Output BAUD Rate:		19200	19200	19200
19		Best Percent Error:		0	0	0
20		Best PLL Multiplier:		**x1**	**x1**	**x1**
21		PLLConfig value:		0x01	0x01	0x01
22		BRGConfig Value:		0x00	0x10	0x20
23		DIVLSB Value:		0x06	0x0C	0x18
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x0A	0x0A	0x0A

Note that cell D19 indicates zero frequency error, at least with regards to the MAX3108 conversion process from frequency input to baud rate output. This is the best scenario, where rate mode is 16. Consult cells D21 through D25 to determine what to program into the MAX3108 clock configuration registers. You will find that, in the majority of the cases, you will select cells D21 though D25 to program the desired baud rate.

An Example with a Frequency Error

Now let's try a different example. Let's generate 190kBd from a 1.8432MHz crystal. Enter the following parameters:

	A	B	C
1	1.21		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	190000
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n

Please note the different result:

	A	B	C	D	E	F	
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**	
14		Flag:		TRUE	TRUE	TRUE	
15		Best Pre-Div:		4	3	4	
16		Best BRG INT:		21	38	87	
17		Best BRG Frac:		13	13	5	
18		Best Output BAUD Rate:		190129.5129	189959.4203	189993.4145	
19		Best Percent Error:		0.068164681	0.021357742	0.003466074	
20		Best PLL Multiplier:		**x144**	**x96**	**x144**	
21		PLLConfig value:		0xC4	0x83	0xC4	
22		BRGConfig Value:		0x0D	0x1D	0x25	
23		DIVLSB Value:		0x15	0x26	0x57	
24		DIVMSB Value:		0x00	0x00	0x00	
25		CLKSource Value:		0x06	0x06	0x06	

Now, we have a small error for a rate mode of 16, and less error for the other rate modes. In this case, the error is small enough that the cell D21 through D25 values should be used.

What is a small enough error? In general, a frequency error of 0.8% or less should be no cause for concern. However, a frequency error of 2.0% or higher should be looked at carefully. Errors between 0.8% and 2.0% involve some judgment. In this case, the frequency error with a rate mode of 16 is about 0.07%, more than small enough to make it the best choice for configuration.

An Example with a Large Frequency Error

We will now try something that shows why we may need to sometimes choose columns E or F. To begin, let's generate 5.678901MBd from a 28.23MHz clock. Enter the following parameters:

	A	B	C	
2		**XIN Frequency:**	1843200	
3		**Desired BAUD Rate:**	5678901	
4				
5		**Crystal (Y/N)?**	n	
6		**CLK on RTS PIN (Y/N)?**	n	
7				

Here, we obtain a more interesting result:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		2	2	2
16		Best BRG INT:		1	1	1
17		Best BRG Frac:		0	15	15
18		Best Output BAUD Rate:		5529600	5707974.194	5707974.194
19		Best Percent Error:		2.629047416	0.511951054	0.511951054
20		Best PLL Multiplier:		**x96**	**x96**	**x48**
21		PLLConfig value:		0x82	0x82	0x42
22		BRGConfig Value:		0x00	0x1F	0x2F
23		DIVLSB Value:		0x01	0x01	0x01
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x04	0x04	0x04

Note the frequency error with a rate mode of 16 is 2.63%, which is too high. If the MAX3108 IRDA feature is used, the rate mode must be set to 16. In this case, perhaps a different input frequency can be chosen, 18MHz for example. Or perhaps a more careful analysis of channel timing can be undertaken, to verify if this error does not impact the specific system being designed.

Another alternative is to choose the configuration with the rate mode set at 8. This results in a very respectable 0.5% frequency error. In this situation, the parameters to program the MAX3108 clock configuration registers would be taken from cells E21 through E25.

An Example with More Limited Choices

For output rates greater than 6MBd, the rate mode cannot be 16. This is another reason why one of the other rate modes must be selected. Even below 6MBd, some combinations of input frequencies and output baud rates conspire to disallow the case where rate mode is 16. As an example, try to generate 5.775MBd from a 1.8432MHz crystal.

	A	B	C
1	1.22		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	5775000
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n

The result is:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		FALSE	TRUE	TRUE
15		Best Pre-Div:			4	4
16		Best BRG INT:			1	2
17		Best BRG Frac:			7	14
18		Best Output BAUD Rate:			5770017.391	5770017.391
19		Best Percent Error:			0.086278938	0.086278938
20		Best PLL Multiplier:			**x144**	**x144**
21		PLLConfig value:			0xC4	0xC4
22		BRGConfig Value:			0x17	0x2E
23		DIVLSB Value:			0x01	0x02
24		DIVMSB Value:			0x00	0x00
25		CLKSource Value:			0x06	0x06

In this case, a rate mode of 16 cannot be used at all. Either the input frequency must be changed, or a rate mode of 8 must be chosen. Note, in some cases, a rate mode of 8 is also not an option. For example, when the output baud rate is greater than 12MBd, the only possible rate mode is 4.

Programming the MAX3108

Once a set of programming parameters has been chosen from the spreadsheet, programming the MAX3108 is a straightforward task. When loading all the registers at power-up, these five registers comprise part of the set being loaded. This section describes how to program the MAX3108 for a target baud rate, presuming there is not yet any serial data flow. Further steps must be taken when changing an existing baud rate. These steps will be explained in a future application note. Let's begin with an example where the frequency source is an external clock, say, generating 190kBd from a 28.23MHz external clock source on the MAX3108 XIN pin. The input section of the spreadsheet would look like:

	A	B	C
1	1.22		
2		**XIN Frequency:**	28230000
3		**Desired BAUD Rate:**	190000
4			
5		**Crystal (Y/N)?**	n
6		**CLK on RTS PIN (Y/N)?**	n

The results are:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		55	37	37
16		Best BRG INT:		24	48	96
17		Best BRG Frac:		5	3	6
18		Best Output BAUD Rate:		190003.2718	190001.0516	190001.0516
19		Best Percent Error:		0.001721996	0.000553492	0.000553492
20		Best PLL Multiplier:		**x144**	**x96**	**x96**
21		PLLConfig value:		0xF7	0xA5	0xA5
22		BRGConfig Value:		0x05	0x13	0x26
23		DIVLSB Value:		0x18	0x30	0x60
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x04	0x04	0x04

Here is the program to load the MAX3108 clock configuration registers to generate the target baud rate. Note that the parameters filled into array U190K in the program are copied from cells D21 through D25 in the spreadsheet.

```
/*
** Run UART at 190KBAUD from a 28.23MHz clock source
*/
unsigned char U190K [] = {
   0xF7, 0x05, 0x18, 0x00, 0x04
} ;

   // Load the MAX3108 clock configuration registers appropriately
   //
   if (!MAX3108_Puts (MAX3108R_PLLCONFIG, sizeof U190K, U190K)) {
      // handle possible I²C protocol error here
   }
```

A Second Programming Example

For another example, let's generate 230.4kBd from a 1.8432MHz crystal as input. This is a very common crystal for UART clock generation. The input section of the spreadsheet would look like:

	A	B	C
1	1.22		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	230400
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n
7			

The results are:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		3	1	1
16		Best BRG INT:		1	1	2
17		Best BRG Frac:		0	0	0
18		Best Output BAUD Rate:		230400	230400	230400
19		Best Percent Error:		0	0	0
20		Best PLL Multiplier:		**x6**	**x1**	**x1**
21		PLLConfig value:		0x03	0x01	0x01
22		BRGConfig Value:		0x00	0x10	0x20
23		DIVLSB Value:		0x01	0x01	0x02
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x06	0x0A	0x0A

When the clock source is a crystal, you have the option of verifying crystal operation. The ClkReady bit in the STSInt register asserts only when the crystal oscillator and the PLL are operating and locked. For a given MAX3108 clock configuration that is known to be working, this reduces to verification that the crystal oscillator is running as expected. The ClkReady bit **does not work** if using an external clock source, so do not test it if an external clock feeds the MAX3108.

Though useful in a development environment, this feature is probably of little use in production or in equipment deployed in the field. Note that other bits in the STSInt register are cleared upon read. There is no way to check the ClkReady bit without also clearing a possible TxEmptyInt or GPIxInt condition. Programmers beware!

In the case of an external clock, if the predivider or PLL factor is changed, the UART will not be ready to transmit or receive immediately. After changing the predivider or PLL factor, or after enabling the PLL, the UART will function at the new programmed baud rate after a time delay of 1200 × (predivider)/(XIN frequency) or less.

```
/*
** Run UART at 230.4KBAUD from a 1.8432MHz crystal
*/
unsigned char U230K4 [] = {
   0x03, 0x00, 0x01, 0x00, 0x06
} ;

   // Load the MAX3108 clock configuration registers appropriately
   //
   if (!MAX3108_Puts (MAX3108R_PLLCONFIG, sizeof U230K4, U230K4)) {
      // handle possible I²C protocol error here
   }

   // Check XTAL and PLL are running OK - optional
   while (MAX3108_Read (MAX3108R_STSINT & 0x0008) == 0) ;
```

Avoiding PLL Changes

So far, we have been using the results from rows 13 through 25. Another section (in rows 27 through 39) has slightly different assumptions. In this second section, configurations with the lowest frequency error are calculated, but with the PLL fixed at different rates.

This comes in handy when generating multiple baud rates from a single frequency source, without incurring the delay associated with PLL relock. This means that the predivider and PLL must not change between the various configurations.

One example of this would be in an IO-Link® scenario, where it is important to be able to change quickly between 230.4kBd 38.4kBd, and 4.8kBd. Let's see how this can be accomplished, assuming our ubiquitous 1.8432MHz crystal. Starting with the highest baud rate, the input section of the spreadsheet would look like:

	A	B	C
1	1.22		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	4800
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n

This results in:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		3	1	1
16		Best BRG INT:		1	1	2
17		Best BRG Frac:		0	0	0
18		Best Output BAUD Rate:		230400	230400	230400
19		Best Percent Error:		0	0	0
20		Best PLL Multiplier:		**x6**	**x1**	**x1**
21		PLLConfig value:		0x03	0x01	0x01
22		BRGConfig Value:		0x00	0x10	0x20
23		DIVLSB Value:		0x01	0x01	0x02
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x06	0x0A	0x0A

Let's start with the assumption that rate mode can be 16. Note the predivider and the PLL factor, 3 and x6, respectively. If this works, we would use the configuration values in cells D21 through D25 to program the MAX3108 for 230.4kBd. Now, let's try the next baud rate.

	A	B	C
1	1.22		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	38400
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n
7			

The result in this case is:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		1	1	1
16		Best BRG INT:		3	6	12
17		Best BRG Frac:		0	0	0
18		Best Output BAUD Rate:		38400	38400	38400
19		Best Percent Error:		0	0	0
20		Best PLL Multiplier:		**x1**	**x1**	**x1**
21		PLLConfig value:		0x01	0x01	0x01
22		BRGConfig Value:		0x00	0x10	0x20
23		DIVLSB Value:		0x03	0x06	0x0C
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x0A	0x0A	0x0A

None of these entries has a PLL factor of x6, so let's now consult the second section.

	A	B	C	D	E	F	G	H
27		**Best, for each PLL Mult**		**x1**	**x6**	**x48**	**x96**	**x144**
28		Flag:		TRUE	TRUE	TRUE	TRUE	TRUE
29		Best Pre-Div:		1	3	2	2	3
30		Best BRG INT:		3	6	72	144	144
31		Best BRG Frac:		0	0	0	0	0
32		Best Output BAUD Rate:		38400	38400	38400	38400	38400
33		Best Percent Error:		0	0	0	0	0
34				**RM = 16**	**RM = 16**	**RM = 16**	**RM = 16**	**RM = 16**
35		PLLConfig value:		0x01	0x03	0x42	0x82	0xC3
36		BRGConfig Value:		0x00	0x00	0x00	0x00	0x00
37		DIVLSB Value:		0x03	0x06	0x48	0x90	0x90
38		DIVMSB Value:		0x00	0x00	0x00	0x00	0x00
39		CLKSource Value:		0x0A	0x06	0x06	0x06	0x06

Column E provides configuration under the assumption that the PLL factor must be x6. Note that the predivider value is also 3. This means that the PLL will not need to relock going from 230.4kBd to 38.4kBd, if we choose to configure

the 38.4kBd output rate with the parameters given in cells E35 through E39.

We need to check that we can do this with 4.8kBd as well. The input section of the spreadsheet would look like:

	A	B	C
1	1.22		
2		**XIN Frequency:**	1843200
3		**Desired BAUD Rate:**	4800
4			
5		**Crystal (Y/N)?**	y
6		**CLK on RTS PIN (Y/N)?**	n

The result is:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		1	1	1
16		Best BRG INT:		24	48	96
17		Best BRG Frac:		0	0	0
18		Best Output BAUD Rate:		4800	4800	4800
19		Best Percent Error:		0	0	0
20		Best PLL Multiplier:		**x1**	**x1**	**x1**
21		PLLConfig value:		0x01	0x01	0x01
22		BRGConfig Value:		0x00	0x10	0x20
23		DIVLSB Value:		0x18	0x30	0x60
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x0A	0x0A	0x0A

Just like in the case of 38.4kBd, none of the results have a PLL factor of x6, so we consult the second section:

	A	B	C	D	E	F	G	H
27		**Best, for each PLL Mult**		**x1**	**x6**	**x48**	**x96**	**x144**
28		Flag:		TRUE	TRUE	TRUE	TRUE	TRUE
29		Best Pre-Div:		1	3	2	2	3
30		Best BRG INT:		24	48	576	1152	1152
31		Best BRG Frac:		0	0	0	0	0
32		Best Output BAUD Rate:		4800	4800	4800	4800	4800
33		Best Percent Error:		0	0	0	0	0
34				**RM = 16**	**RM = 16**	**RM = 16**	**RM = 16**	**RM = 16**
35		PLLConfig value:		0x01	0x03	0x42	0x82	0xC3
36		BRGConfig Value:		0x00	0x00	0x00	0x00	0x00
37		DIVLSB Value:		0x18	0x30	0x40	0x80	0x80
38		DIVMSB Value:		0x00	0x00	0x02	0x04	0x04
39		CLKSource Value:		0x0A	0x06	0x06	0x06	0x06

And just like in the case of 38.4kBd, the x6 section shows we can generate 4.8kBd with a PLL factor of x6 and a predivider of 3, using the configuration parameters found in cells E35 through E39.

Avoiding the PLL Altogether

It is best to avoid the PLL altogether in some scenarios. For example, the PLL does not function if the V_{CC} supplied to the MAX3108 is below 2.35V. Also, if it is important to operate quickly after coming out of a low power mode, the PLL should be avoided. Suppose we need to generate 230.4kBd, 38.4kBd, and 4.8kBd from a 1.8432MHz crystal, but we also need to avoid using the PLL.

The easiest way to accomplish this would be to consult the section with the PLL factor fixed at x1 for all three instances of the spreadsheet. For a target baud rate of 230.4kBd, the second section is:

	A	B	C	D	E	F	G	H
27		**Best, for each PLL Mult**		**x1**	**x6**	**x48**	**x96**	**x144**
28		Flag:		TRUE	TRUE	TRUE	TRUE	TRUE
29		Best Pre-Div:		1	3	2	2	3
30		Best BRG INT:		1	1	12	24	24
31		Best BRG Frac:		0	0	0	0	0
32		Best Output BAUD Rate:		230400	230400	230400	230400	230400
33		Best Percent Error:		0	0	0	0	0
34				**RM = 8**	**RM = 16**	**RM = 16**	**RM = 16**	**RM = 16**
35		PLLConfig value:		0x01	0x03	0x42	0x82	0xC3
36		BRGConfig Value:		0x10	0x00	0x00	0x00	0x00
37		DIVLSB Value:		0x01	0x01	0x0C	0x18	0x18
38		DIVMSB Value:		0x00	0x00	0x00	0x00	0x00
39		CLKSource Value:		0x0A	0x06	0x06	0x06	0x06

Cells D35 through D39 report the required configuration for all three instances of the spreadsheet, one instance for each of the three desired baud rates. Since the PLL is not used in any of these three configurations, and since the predivider value does not change, switching between all three baud rates avoids delays from PLL relock.

Keeping the Rate Mode High

If it is desired to keep the rate mode set at 16 and also avoid the PLL, the third results section, rows 41 through 53, should be consulted. Note that this is a fairly restrictive scenario, so this third section will often be blank to indicate that no such configuration can be reasonably achieved. As an example, let's generate a 190kBd output rate from a 28.32MHz external clock source.

The input section of the spreadsheet is:

	A	B	C
1	1.22		
2		**XIN Frequency:**	28230000
3		**Desired BAUD Rate:**	190000
4			
5		**Crystal (Y/N)?**	n
6		**CLK on RTS PIN (Y/N)?**	n

The result in this case is:

	A	B	C	D	E	F
13		**Best, for each Rate Mode**		**RM = 16**	**RM = 8**	**RM = 4**
14		Flag:		TRUE	TRUE	TRUE
15		Best Pre-Div:		55	37	37
16		Best BRG INT:		24	48	96
17		Best BRG Frac:		5	3	6
18		Best Output BAUD Rate:		190003.2718	190001.0516	190001.0516
19		Best Percent Error:		0.001721996	0.000553492	0.000553492
20		Best PLL Multiplier:		**x144**	**x96**	**x96**
21		PLLConfig value:		0xF7	0xA5	0xA5
22		BRGConfig Value:		0x05	0x13	0x26
23		DIVLSB Value:		0x18	0x30	0x60
24		DIVMSB Value:		0x00	0x00	0x00
25		CLKSource Value:		0x04	0x04	0x04

For a rate mode of 16, the suggested configuration uses a PLL factor of x144, but we need a PLL factor of x1. This does not meet our requirement of a rate mode of 16 and a PLL factor of x1. Consulting the second results section:

	A	B	C	D	E	F	G	H
27		**Best, for each PLL Mult**		**x1**	**x6**	**x48**	**x96**	**x144**
28		Flag:		TRUE	TRUE	TRUE	TRUE	TRUE
29		Best Pre-Div:		1	41	25	37	44
30		Best BRG INT:		18	5	71	48	121
31		Best BRG Frac:		9	7	5	3	9
32		Best Output BAUD Rate:		190101.0101	189941.127	190014.3734	190001.0516	190003.272
33		Best Percent Error:		0.053163211	0.030985791	0.007564925	0.000553492	0.001722
34				**RM = 8**	**RM = 4**	**RM = 4**	**RM = 8**	**RM = 4**
35		PLLConfig value:		0x01	0x29	0x59	0xA5	0xEC
36		BRGConfig Value:		0x19	0x27	0x25	0x13	0x29
37		DIVLSB Value:		0x12	0x05	0x47	0x30	0x79
38		DIVMSB Value:		0x00	0x00	0x00	0x00	0x00
39		CLKSource Value:		0x08	0x04	0x04	0x04	0x04

For a PLL factor of x1, the suggested rate mode is 8. Not quite what we are looking for. Consulting the third results section of the spreadsheet:

	A	B	C	D
41		**Best if no PLL and RM = 16**		
42		Flag:		TRUE
43		Best Pre-Div:		1
44		Best BRG INT:		9
45		Best BRG Frac:		5
46		Best Output BAUD Rate:		189463.0872
47		Best Percent Error:		0.282585659
48				
49		PLLConfig value:		0x01
50		BRGConfig Value:		0x05
51		DIVLSB Value:		0x09
52		DIVMSB Value:		0x00
53		CLKSource Value:		0x08

We see that generating 190kBd from a 28.23MHz external clock is possible, with a PLL factor of x1 and a rate mode of 16. The frequency error is a reasonable 0.28%. The configuration parameters to accomplish this are in cells D49 through D53.

The Manual Tab

For experimentation, or when dealing with unusual situations, the **Manual** tab of the spreadsheet provides some basic assistance. The spreadsheet has four sections. The user input section is:

	A	B	C	D	E	F	G	H	I
3		**User Input Section:**							
4		Fin:	28230000						
5		Desired BAUD Rate:	190000						
6		Crystal? (Y, N):	n			Input is Crystal? If not, must be clock on XIN pin			
7		CLK-ro-RTS Pin? (Y, n):	n						
8		Test Pre-Div (1-63):	1						
9		Test PLL (1,6,48,96,144):	1						
10		Test BRG Int (1-65535):	9						
11		Test BRG Frac: (0-15):	5						
12		Test Rate Mode (16,8,4):	16						

After entering all these parameters, consult the second section:

	A	B	C	D	E	F	G	H	I	J	K
14		**Validations and tests:**									
15		XIN Test (XTAL):	TRUE			Crystal frquency must be between 1MHz and 4MHz					
16		XIN Test (EXT OSC):	TRUE	GOOD		External clock frequency must be between 0.5MHz and 35MHz					
17		Pre-Div Test:	TRUE	GOOD		Pre-div must be between 1 and 63					
18		PLL Test:	TRUE	GOOD		PLL must be one of 1, 6, 48, 96, 144					
19		PLL Frequency Input:	28230000								
20		PLL x1 Range Test:	TRUE	GOOD		PLL Bypassed, freq must be under 96MHz					
21		PLL x6 Range Test:	TRUE			PLL x6, freq must be between 500KHz and 800KHz					
22		PLL x48 Range Test:	TRUE			PLL x48, freq must be between 850KHz and 1.2MHz					
23		PLL x96 Range Test:	TRUE			PLL x96, freq must be between 425KHz and 1MHz					
24		PLL x144 Range Test:	TRUE			PLL x144, freq must be between 390KHz and 667KHz					
25		PLL Frequency Output:	28230000								
26		BRG INT part Test:	TRUE			Integer part must be between 1 and 65,535					
27		BRG FRAC part Test:	TRUE	GOOD		Fractional Part must be between 0 and 15					
28		BRG Freq Output:	3031409								
29		Rate Mode Test:	TRUE	GOOD		Rate Mode must be one of 16, 8, or 4					

Make sure that no cell in column D reports **FALSE**, and that no cell in column D reports **BAD**. Correct any errors in the user input section before proceeding any further. Once all errors are cleared, consult the third section:

	A	B	C	D
32		**Outputs:**		
33		BAUD Rate:	189463	OK
34		Frequency Error Percentage:	-0.28258566	
35		PLLConfig value:	0x01	
36		BRGConfig Value:	0x05	
37		DIVLSB Value:	0x09	
38		DIVMSB Value:	0x00	
39		CLKSource Value:	0x08	

If you see **WARNING** in cell D33, you need to go back and fix an error in your input. Otherwise the actual baud rate generated appears in cell C33, the percent error as compared to the desired baud rate (cell C5) appears in cell C34, and the configuration parameters to program the MAX3108 clock configuration registers appear in cells C35 through

C39. Note: unless you have had some assistance, the frequency error could be quite large, which is the reason for the fourth section:

	A	B	C	D	E	F	G	H
41		**Desired BAUD Rate Calcs:**				Figures BRG INT and FRAC part only		
42		Desired BAUD Rate:	190000			**USER INPUT HERE!**		
43		BRG INT to use:	9					
44		BRG FRAC to use:	5					
45		Actual Frequency:	189463					
46		Frequency Error Percentage:	-0.28258566					
47								

Here, the only input is the desired baud rate in cell C42. It assumes a predivider, PLL factor, and rate mode as entered in the user input section, and calculates the best baud rate generator parameters to use. To make use of these, copy them back to the user input section, manually copying cell C43 to cell C10, and cell C44 to cell C11. Otherwise, the results of this fourth section are not reflected in the third section, the outputs section.

A Great Debug Trick

The MAX3108 provides a way to verify what the output baud rate is. This feature is useful during development, debugging, and possibly during manufacture. When this feature is activated, the MAX3108 feeds the output of the baud rate generator, just before the rate mode divider, to the RTS pin.

For example, let's configure the MAX3108 for a baud rate of 230.4kBd, using a rate mode of 16. Activating this feature, the RTS pin will supply a clock with a frequency of 230.4K × 16, or 3.6864MHz. This can be verified attaching a frequency counter to the RTS pin, or approximately, with an oscilloscope. This feature is enabled when the CLKtoRTS bit, bit 7, in the CLKSource register is set to 1. This is what that parameter does in the **Synthesize** tab of the spreadsheet:

	A	B	C
6		**CLK on RTS PIN (Y/N)?**	n

Setting this input to y causes all reported configurations to enable this feature. Once loaded into the MAX3108, these configurations will supply a clock to the RTS pin. This is one way to take advantage of this feature.

Another method is to turn the feature on or off without changing the clock configuration already programmed into the MAX3108. The following code snippet shows how this might be done. These functions would turn on or off the feature without otherwise disturbing the clock configuration.

```
/*
** Function to enable the clock-to-RTS feature of the MAX3108
**
** Returns TRUE if operation successfully completed.
*/
BOOL CLKtoRTS_En (void) {
   unsigned int temp;
```

```
   temp = MAX3108_Read (MAX3108R_CLKSOURCE);  // returns 0xff00 if I²C error
   if (temp & 0xff00)  return FALSE;
   return MAX3108_Write (MAX3108R_CLKSOURCE, temp | 0x0080);
}

/*
** Function to disable the clock-to-RTS feature of the MAX3108
**
** Returns TRUE if operation completed successfully
*/
BOOL CLKtoRTS_Dis (void) {
   unsigned int temp;

   temp = MAX3108_Read (MAX3108R_CLKSOURCE);  // returns 0xff00 if I²C error
   if (temp & 0xff00)  return FALSE;
   return MAX3108_Write (MAX3108R_CLKSOURCE, temp & 0x007f);
}
```

Conclusion

This application note has shown how to use a spreadsheet to quickly and easily calculate the parameters needed to program the MAX3108 for any desired baud rate. It also explains how to program the MAX3108 using these parameters.

Appendix

This appendix contains the definitions necessary to understand the pseudocode.

Definitions Relating to Pin I/O

POLL_MAX3108_IRQ: This pseudocode returns the digital state of the MAX3108 IRQ pin. This pin requires an external pullup to work properly.

SET_MAX3108_CS_PIN_HIGH: This pseudocode drives the MAX3108 CS/A0 pin high.

SET_MAX3108_CS_PIN_LOW: This pseudocode drives the MAX3108 CS/A0 pin low.

SET_MAX3108_RESET_PIN_HIGH: This pseudocode drives the MAX3108 RST pin high.

SET_MAX3108_RESET_PIN_LOW: This pseudocode drives the MAX3108 RST pin low.

WAIT: This pseudocode delays execution of the following instruction by at least the specified time.

Definitions Relating to the I²C Interface

I2C_SEND_BYTE: This pseudocode sends one byte to the MAX3108 via the I²C bus.

I2C_SET_RESTART_CONDITION: This pseudocode puts a restart condition on the I²C bus.

I2C_SET_START_CONDITION: This pseudocode puts a start condition on the I²C bus.

I2C_SET_STOP_CONDITION: This pseudocode puts a stop condition on the I²C bus.

I2C_TEST_ACK: This pseudocode returns TRUE if the MAX3108 responded to a byte written with an ACK, and returns FALS if the MAX3108 responded to a byte written with a NACK.

SPI_SEND_BYTE: One byte is sent to the MAX3108 via the SPI interface, and the byte returned by the MAX3108 at the same time is captured.

Definitions Related to the Register Interface

MAX3108_Gets: This routine is either MAX3108_I2C_Gets or MAX3108_SPI_Gets, depending on which interface is implemented in the application.

MAX3108_Puts: This routine is either MAX3108_I2C_Puts or MAX3108_SPI_Puts, depending on which interface is implemented in the application.

MAX3108_Read: This routine is either MAX3108_I2C_Read or MAX3108_SPI_Read, depending on which interface is implemented in the application.

MAX3108_Write: This routine is either MAX3108_I2C_Write or MAX3108_SPI_Write, depending on which interface is implemented in the application.

MAX3108_I2C_Gets: This routine burst reads from the MAX3108 receive FIFO or other MAX3108 registers via the I²C interface.

MAX3108_I2C_Puts: This routine burst writes to the MAX3108 transmit FIFO or other MAX3108 registers via the I²C interface.

MAX3108_I2C_Read: This routine reads the value of one of the MAX3108 registers via the I²C interface.

MAX3108_I2C_Write: This routine writes a value to one of the MAX3108 registers via the I²C interface.

MAX3108_SPI_Gets: This routine burst reads from the MAX3108 receive FIFO or other MAX3108 registers via the SPI interface.

MAX3108_SPI_Puts: This routine burst writes to the MAX3108 transmit FIFO or other MAX3108 registers via the SPI interface.

MAX3108_SPI_Read: This routine reads the value of one of the MAX3108 registers via the SPI interface.

MAX3108_SPI_Write: This routine writes a value to one of the MAX3108 registers via the SPI interface.

MAX3108 Register Defines

```
//
// MAX3108 Register map defines
//
#define MAX3108R_RHR          (0x00)
```

```
#define MAX3108R_THR          (0x00)
#define MAX3108R_IRQEN        (0x01)
#define MAX3108R_ISR          (0x02)
#define MAX3108R_LSRINTEN     (0x03)
#define MAX3108R_LSR          (0x04)
#define MAX3108R_SPCLCHRINTEN (0x05)
#define MAX3108R_SPCLCHARINT  (0x06)
#define MAX3108R_STSINTEN     (0x07)
#define MAX3108R_STSINT       (0x08)
#define MAX3108R_MODE1        (0x09)
#define MAX3108R_MODE2        (0x0a)
#define MAX3108R_LCR          (0x0b)
#define MAX3108R_RXTIMEOUT    (0x0c)
#define MAX3108R_HDPLXDELAY   (0x0d)
#define MAX3108R_IRDA         (0x0e)
#define MAX3108R_FLOWLVL      (0x0f)
#define MAX3108R_FIFOTRGLVL   (0x10)
#define MAX3108R_TXFIFOLVL    (0x11)
#define MAX3108R_RXFIFOLVL    (0x12)
#define MAX3108R_FLOWCTRL     (0x13)
#define MAX3108R_XON1         (0x14)
#define MAX3108R_XON2         (0x15)
#define MAX3108R_XOFF1        (0x16)
#define MAX3108R_XOFF2        (0x17)
#define MAX3108R_GPIOCONFG    (0x18)
#define MAX3108R_GPIODATA     (0x19)
#define MAX3108R_PLLCONFIG    (0x1a)
#define MAX3108R_BRGCONFIG    (0x1b)
#define MAX3108R_DIVLSB       (0x1c)
#define MAX3108R_DIVMSB       (0x1d)
#define MAX3108R_CLKSOURCE    (0x1e)
```


Related Parts	
MAX3108	SPI/I²C UART with 128-Word FIFOs in WLP

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5306: http://www.maximintegrated.com/an5306