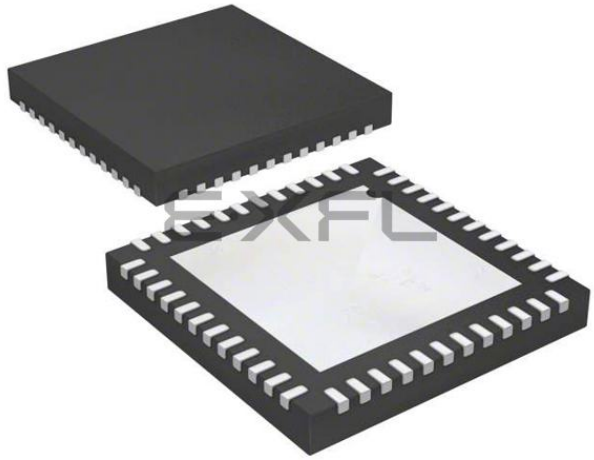

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	ARM7®
Core Size	16/32-Bit
Speed	20.48MHz
Connectivity	LINbus, SPI, UART/USART
Peripherals	PSM, Temp Sensor, WDT
Number of I/O	9
Program Memory Size	96KB (48K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1.5K x 32
Voltage - Supply (Vcc/Vdd)	3.5V ~ 18V
Data Converters	A/D 2x16b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 115°C (TA)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad, CSP
Supplier Device Package	48-LFCSP-VQ (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/aduc7036bcpz

Table 3. SPI Master Mode—Phase Mode = 0

Parameter	Description	Min	Typ	Max	Unit
t_{SL}	SCLK low pulse width[1]		(SPIDIV + 1) × t_{HCLK}		ns
t_{SH}	SCLK high pulse width[1]		(SPIDIV + 1) × t_{HCLK}		ns
t_{DAV}	Data output valid after SCLK edge[2]			(2 × t_{UCLK}) + (2 × t_{HCLK})	ns
t_{DOSU}	Data output setup before SCLK edge		0.5 t_{SL}		ns
t_{DSU}	Data input setup time before SCLK edge	0			ns
t_{DHD}	Data input hold time after SCLK edge[2]	3 × t_{UCLK}			ns
t_{DF}	Data output fall time		3.5		ns
t_{DR}	Data output rise time		3.5		ns
t_{SR}	SCLK rise time		3.5		ns
t_{SF}	SCLK fall time		3.5		ns

[1] t_{HCLK} depends on the clock divider (CD) bits in the POWCON MMR. $t_{HCLK} = t_{UCLK}/2^{CD}$.
[2] t_{UCLK} = 48.8 ns. It corresponds to the 20.48 MHz internal clock from the PLL before the clock divider.

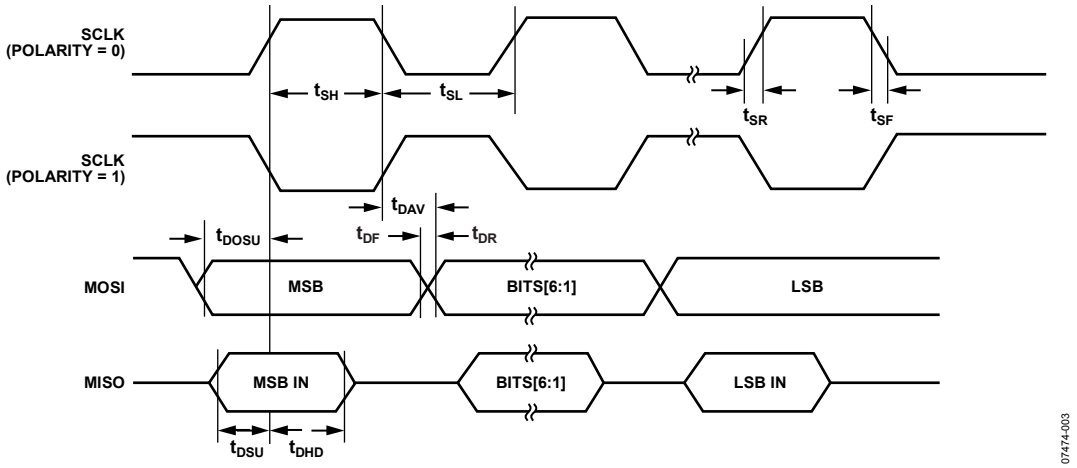

Figure 3. SPI Master Mode Timing—Phase Mode = 0

LIN Timing Specifications

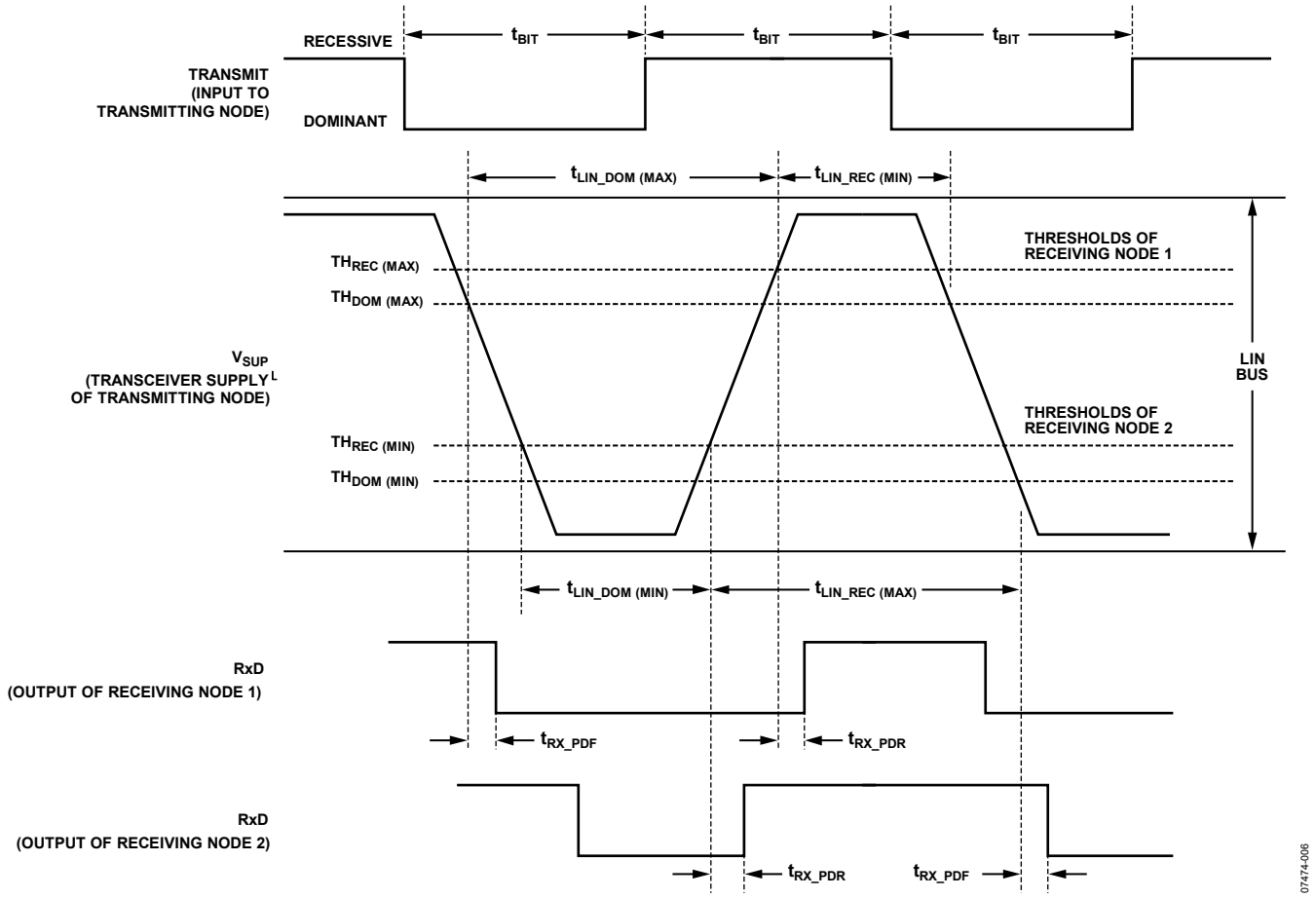

Figure 6. LIN 2.0 Timing Specification

ARM7 Exceptions

The ARM7 supports five types of exceptions with a privileged processing mode associated with each type. The five types of exceptions are as follows:

- Normal interrupt or IRQ. This is provided to service general-purpose interrupt handling of internal and external events.
- Fast interrupt or FIQ. This is provided to service data transfer or a communication channel with low latency. FIQ has priority over IRQ.
- Memory abort (prefetch and data).
- Attempted execution of an undefined instruction.
- Software interrupt (SWI) instruction that can be used to make a call to an operating system.

Typically, the programmer defines interrupts as IRQ, but for higher priority interrupts, the programmer can define interrupts as the FIQ type.

The priority of these exceptions and vector address are listed in Table 9.

Table 9. Exception Priorities and Vector Addresses

Priority	Exception	Address
1	Hardware reset	0x00
2	Memory abort (data)	0x10
3	FIQ	0x1C
4	IRQ	0x18
5	Memory abort (prefetch)	0x0C
6	Software interrupt[1]	0x08
6	Undefined instruction[1]	0x04

[1] A software interrupt and an undefined instruction exception have the same priority and are mutually exclusive.

The list of exceptions in Table 9 are located from 0x00 to 0x1C, with a reserved location at 0x14. This location is required to be written with either 0x27011970 or the checksum of Page 0, excluding Location 0x14. If this is not done, user code does not execute and LIN download mode is entered.

ARM Registers

The ARM7TDMI has 16 standard registers. R0 to R12 are used for data manipulation, R13 is the stack pointer, R14 is the link register, and R15 is the program counter that indicates the instruction currently being executed. The link register contains the address from which the user has branched (if the branch and link command was used) or the command during which an exception occurred.

The stack pointer contains the current location of the stack. As a general rule, on an ARM7TDMI, the stack starts at the top of the available RAM area and descends using the area as required. A separate stack is defined for each of the exceptions. The size of each stack is user configurable and is dependent on the target application. On the ADuC7036, the stack begins at 0x00040FFC and descends. When programming using high level languages, such as C, it is necessary to ensure that the stack does not overflow. This is dependent on the performance of the compiler that is used.

When an exception occurs, some of the standard registers are replaced with registers specific to the exception mode. All exception modes have replacement banked registers for the stack pointer (R13) and the link register (R14) as represented in Figure 11. The FIQ mode has more registers (R8 to R12) supporting faster interrupt processing. With the increased number of noncritical registers, the interrupt can be processed without the need to save or restore these registers, thereby reducing the response time of the interrupt handling process.

More information relative to the model of the programmer and the ARM7TDMI core architecture can be found in ARM7TDMI technical and ARM architecture manuals available directly from ARM Ltd.

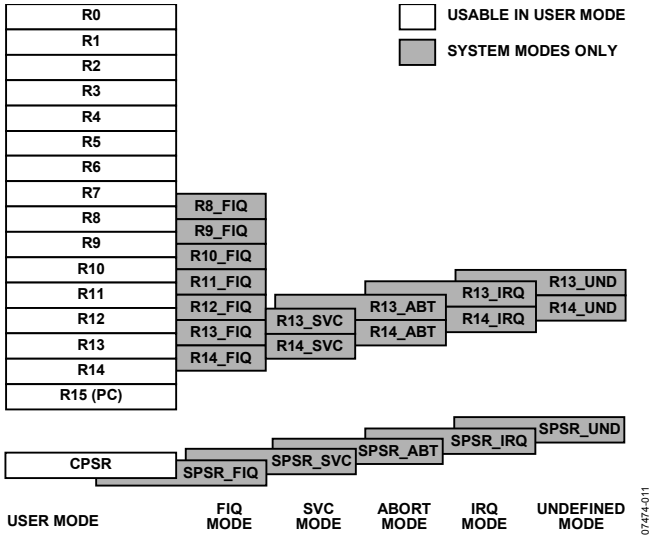

Figure 11. Register Organization

Interrupt Latency

The worst-case latency for an FIQ consists of the longest possible time for the request to pass through the synchronizer, for the longest instruction to complete (the longest instruction is an LDM) and load all the registers including the PC, and for the data abort entry and the FIQ entry to complete. At the end of this time, the ARM7TDMI executes the instruction at Address 0x1C (the FIQ interrupt vector address). The maximum FIQ latency is 50 processor cycles or just over 2.44 µs in a system using a continuous 20.48 MHz processor clock.

The maximum IRQ latency calculation is similar but must allow for the fact that FIQ has higher priority and may delay entry into the IRQ handling routine for an arbitrary length of time. This time can be reduced to 42 cycles if the LDM command is not used; some compilers have an option to compile without using this command. Another option is to run the part in Thumb mode, which reduces the time to 22 cycles.

Remap Operation

When a reset occurs on the ADuC7036, execution starts automatically in the factory-programmed internal configuration code. This so-called kernel is hidden and cannot be accessed by user code. If the ADuC7036 is in normal mode, it executes the power-on configuration routine of the kernel and then jumps to the reset vector, Address 0x00000000, to execute the reset exception routine of the user. Because the Flash/EE is mirrored at the bottom of the memory array at reset, the reset routine must always be written in Flash/EE.

The remap command must be executed from the absolute Flash/EE address and not from the mirrored, remapped segment of memory, which may be replaced by SRAM. If a remap operation is executed while operating code from the mirrored location, prefetch/data aborts may occur or the user may observe abnormal program operation.

Any kind of reset remaps the Flash/EE memory to the bottom of the memory array.

SYSMAP0 Register

Name: SYSMAP0

Address: 0xFFFF0220

Default Value: Updated by the kernel

Access: Read/write access

Function: This 8-bit register allows user code to remap either RAM or Flash/EE space into the bottom of the ARM memory space, starting at Address 0x00000000.

Table 10. SYSMAP0 MMR Bit Designations

Bit	Description
7 to 1	Reserved. These bits are reserved and should be written as 0 by user code.
0	Remap bit. Set by the user to remap the SRAM to 0x00000000. Cleared automatically after a reset to remap the Flash/EE memory to 0x00000000.

The FEE0CON and FEE1CON Registers section to the FEE0MOD and FEE1MOD Registers section provide detailed descriptions of the bit designations for each of the Flash/EE control MMRs.

FEE0CON and FEE1CON Registers

Name: FEE0CON and FEE1CON

Address: 0xFFFF0E08 and 0xFFFF0E88

Default Value: 0x07

Access: Read/write access

Function: These 8-bit registers are written by user code to control the operating modes of the Flash/EE memory controllers for Block 0 (32 kB) and Block 1 (64 kB).

Table 13. Command Codes in FEE0CON and FEE1CON

Code	Command	Description[1]
0x00[2]	Reserved	Reserved. This command should not be written by user code.
0x01[2]	Single read	Load FEExDAT with the 16-bit data indexed by FEExADR.
0x02[2]	Single write	Write FEExDAT at the address pointed by FEExADR. This operation takes 50 µs.
0x03[2]	Erase write	Erase the page indexed by FEExADR and write FEExDAT at the location pointed by FEExADR. This operation takes 20 ms.
0x04[2]	Single verify	Compare the contents of the location pointed by FEExADR to the data in FEExDAT. The result of the comparison is returned in FEExSTA, Bit 1 or Bit 0.
0x05[2]	Single erase	Erase the page indexed by FEExADR.
0x06[2]	Mass erase	Erase Block 0 (32 kB) or Block 1 (64 kB) of user space. The 2 kB kernel is protected. This operation takes 1.2 sec. To prevent accidental execution, a command sequence is required to execute this instruction (see the Command Sequence for Executing a Mass Erase section).
0x07		Default command.
0x08	Reserved	Reserved. This command should not be written by user code.
0x09	Reserved	Reserved. This command should not be written by user code.
0x0A	Reserved	Reserved. This command should not be written by user code.
0x0B	Signature	FEE0CON: This command results in the generation of a 24-bit linear feedback shift register (LFSR)-based signature that is loaded into FEE0SIG. If FEE0ADR is less than 0x97800, this command results in a 24-bit LFSR-based signature of the user code space from the page specified in FEE0ADR upwards, including the kernel, security bits, and Flash/EE key. If FEE0ADR is greater than 0x97800, the kernel and manufacturing data are signed. This operation takes 120 µs. FEE1CON: This command results in the generation of a 24-bit LFSR-based signature, beginning at FEE1ADR and ending at the end of the 63,500 block, that is loaded into FEE1SIG. The last page of this block is not included in the sign generation.
0x0C	Protect	This command can be run only once. The value of FEExPRO is saved and can be removed only with a mass erase (0x06) or with the software protection key.
0x0D	Reserved	Reserved. This command should not be written by user code.
0x0E	Reserved	Reserved. This command should not be written by user code.
0x0F	Ping	No operation, interrupt generated.

[1] The x represents 0 or 1, designating Flash/EE Block 0 or Block 1.
[2] The FEE0CON register reads 0x07 immediately after the execution of this command.

ADC MMR INTERFACE

The ADC is controlled and configured using several MMRs that are described in detail in the ADC Status Register section to the Low Power Voltage Reference Scaling Factor section.

All bits defined in the top eight MSBs (Bits[8:15]) of the ADCSTA MMR are used as flags only and do not generate interrupts. All bits defined in the lower eight LSBs (Bits[0:7]) of this MMR are logic OR'ed to produce a single ADC interrupt to the MCU core. In response to an ADC interrupt, user code should interrogate the ADCSTA MMR to determine the source of the interrupt. Each ADC interrupt source can be individually masked via the ADCMSKI MMR described in the ADC Interrupt Mask Register section.

All ADC result ready bits are cleared by a read of the ADC0DAT MMR. If the current channel ADC is not enabled, all ADC result ready bits are cleared by a read of the ADC1DAT or ADC2DAT MMRs. To ensure that I-ADC and V-/T-ADC conversion data are synchronous, user code should first read the ADC1DAT MMR and then the ADC0DAT MMR. New ADC conversion results are not written to the ADCxDAT MMRs unless the respective ADC result ready bits are first cleared. The only exception to this rule is the data conversion result updates when the ARM core is powered down. In this mode, ADCxDAT registers always contain the most recent ADC conversion result, even though the ready bits have not been cleared.

ADC Status Register

Name: ADCSTA

Address: 0xFFFF0500

Default Value: 0x0000

Access: Read only

Function: This read only register holds general status information related to the mode of operation or current status of the ADCs.

Table 35. ADCSTA MMR Bit Designations

Bit	Description
15	ADC calibration status. Set automatically in hardware to indicate that an ADC calibration cycle has been completed. Cleared after ADCMDE is written to.
14	ADC temperature conversion error. Set automatically in hardware to indicate that a temperature conversion overrange or underrange has occurred. The conversion result is clamped to negative full scale (underrange error) or positive full scale (overrange error) in this case. Cleared when a valid (in-range) temperature conversion result is written to the ADC2DAT register.
13	ADC voltage conversion error. Set automatically in hardware to indicate that a voltage conversion overrange or underrange has occurred. The conversion result is clamped to negative full scale (underrange error) or positive full scale (overrange error) in this case. Cleared when a valid (in-range) voltage conversion result is written to the ADC1DAT register.
12	ADC current conversion error. Set automatically in hardware to indicate that a current conversion overrange or underrange has occurred. The conversion result is clamped to negative full scale (underrange error) or positive full scale (overrange error) in this case. Cleared when a valid (in-range) current conversion result is written to the ADC0DAT register.
11 to 5	Not used. These bits are reserved for future functionality and should not be monitored by user code.
4	Current channel ADC comparator threshold. Valid only if the current channel ADC comparator is enabled via the ADCCFG MMR. Set by hardware if the absolute value of the I-ADC conversion result exceeds the value written in the ADC0TH MMR. However, if the ADC threshold counter is used (ADC0TCL), this bit is set only when the specified number of I-ADC conversions equals the value in the ADC0THV MMR. Cleared automatically by hardware when reconfiguring the ADC or if the comparator is disabled.
3	Current channel ADC overrange bit. Set by hardware if the overrange detect function is enabled via the ADCCFG MMR and the I-ADC input is grossly (>30% approximate) over range. This bit is updated every 125 µs. Cleared by software only when ADCCFG[2] is cleared to disable the function, or the ADC gain is changed via the ADC0CON MMR.
2	Temperature conversion result ready bit. Set by hardware, if the temperature channel ADC is enabled, as soon as a valid temperature conversion result is written in the temperature data register (ADC2DAT MMR). It is also set at the end of a calibration. Cleared by reading either ADC2DAT or ADC0DAT.
1	Voltage conversion result ready bit. Set by hardware, if the voltage channel ADC is enabled, as soon as a valid voltage conversion result is written in the voltage data register (ADC1DAT MMR). It is also set at the end of a calibration. Cleared by reading either ADC1DAT or ADC0DAT.
0	Current conversion result ready bit. Set by hardware, if the current channel ADC is enabled, as soon as a valid current conversion result is written in the current data register (ADC0DAT MMR). It is also set at the end of a calibration. Cleared by reading ADC0DAT.

ADC Interrupt Mask Register

Name: ADCMSKI
Address: 0xFFFF0504
Default Value: 0x00
Access: Read/write
Function: This register allows the ADC interrupt sources to be individually enabled. The bit positions in this register are the same as the lower eight bits in the ADCSTA MMR. If a bit is set by user code to 1, the respective interrupt is enabled. By default, all bits are 0, meaning all ADC interrupt sources are disabled.

ADC Mode Register

Name: ADCMDE
Address: 0xFFFF0508
Default Value: 0x00
Access: Read/write
Function: This 8-bit register configures the mode of operation of the ADC subsystem.

Table 36. ADCMDE MMR Bit Designations

Bit	Description
7	Not used. This bit is reserved for future functionality and should be written as 0 by user code.
6	20 kΩ resistor select. Set to 1 to select the 20 kΩ resistor as shown in Figure 21. Set to 0 to select the direct path to ground as shown in Figure 21 (default).
5	Low power mode reference select. Set to 1 to enable the precision voltage reference in either low power mode or low power plus mode, thereby increasing current consumption. Set to 0 to enable the low power voltage reference in either low power mode or low power plus mode (default).
4 to 3	ADC power mode configuration. 00 = ADC normal mode. If enabled, the ADC operates with normal current consumption yielding optimum electrical performance. 01 = ADC low power mode. If enabled, the I-ADC operates with reduced current consumption. This limitation in current consumption is achieved (at the expense of ADC noise performance) by fixing the gain to 128 and using the on-chip low power (131 kHz) oscillator to directly drive the ADC circuits. 10 = ADC low power plus mode. If enabled, the ADC operates with reduced current consumption. In this mode, the gain is fixed to 512 and the current consumed is approximately 200 μA more than the ADC low power mode. The additional current consumed also ensures that the ADC noise performance is better than that achieved in ADC low power mode. 11 = not defined.
2 to 0	ADC operation mode configuration. 000 = ADC power-down mode. All ADC circuits (including internal reference) are powered down. 001 = ADC continuous conversion mode. In this mode, any enabled ADC continuously converts. 010 = ADC single conversion mode. In this mode, any enabled ADC performs a single conversion. The ADC enters idle mode when the single shot conversion is complete. A single conversion takes two to three ADC clock cycles depending on the chop mode. 011 = ADC idle mode. In this mode, the ADC is fully powered on but is held in reset. 100 = ADC self-offset calibration. In this mode, an offset calibration is performed on any enabled ADC using an internally generated 0 V. The calibration is carried out at the user programmed ADC settings; therefore, as with a normal single ADC conversion, it takes two to three ADC conversion cycles before a fully settled calibration result is ready. The calibration result is automatically written to the ADCxOF MMR of the respective ADC. The ADC returns to idle mode and the calibration and conversion ready status bits are set at the end of an offset calibration cycle. 101 = ADC self-gain calibration. In this mode, a gain calibration against an internal reference voltage is performed on all enabled ADCs. A gain calibration is a two-stage process and takes twice the time of an offset calibration. The calibration result is automatically written to the ADCxGN MMR of the respective ADC. The ADC returns to idle mode, and the calibration and conversion ready status bits are set at the end of a gain calibration cycle. An ADC self-gain calibration should only be carried out on the current channel ADC. Preprogrammed, factory calibration coefficients (downloaded automatically from internal Flash/EE) should be used for voltage temperature measurements. If an external NTC is used, an ADC self-calibration should be performed on the temperature channel. 110 = ADC system zero-scale calibration. In this mode, a zero-scale calibration is performed on enabled ADC channels against an external zero-scale voltage driven at the ADC input pins. The calibration is carried out at the user programmed ADC settings; therefore, as with a normal, single ADC conversion, it takes three ADC conversion cycles before a fully settled calibration result is ready. 111 = ADC system full-scale calibration. In this mode, a full-scale calibration is performed on enabled ADC channels against an external full-scale voltage driven at the ADC input pins.

Current Channel ADC Control Register

Name: ADC0CON

Address: 0xFFFF050C

Default Value: 0x0000

Access: Read/write

Function: This 16-bit register is used to configure the I-ADC.

Note that if the current ADC is reconfigured via ADC0CON, the voltage ADC and temperature ADC are also reset.

Table 37. ADC0CON MMR Bit Designations

Bit	Description
15	Current channel ADC enable. Set to 1 by user code to enable the I-ADC. Cleared to 0 to power down the I-ADC and reset the respective ADC ready bit in the ADCSTA MMR to 0.
14, 13	IIN current source enable. 00 = current sources off. 01 = enables the 50 µA current source on IIN+. 10 = enables the 50 µA current source on IIN−. 11 = enables the 50 µA current source on both IIN− and IIN+.
12 to 10	Not used. These bits are reserved for future functionality and should be written as 0.
9	Current channel ADC output coding. Set to 1 by user code to configure I-ADC output coding as unipolar. Cleared to 0 by user code to configure I-ADC output coding as twos complement.
8	Not used. This bit is reserved for future functionality and should be written as 0.
7, 6	Current channel ADC input select. 00 = IIN+, IIN− are selected. 01 = IIN−, IIN− are selected. Diagnostic, internal short configuration. 10 = V_{REF}/136, 0 V, diagnostic, test voltage for gain settings ≤ 128. Note that if (REG_AVDD, AGND) divided-by-2 reference is selected, REG_AVDD is used for VREF in this mode. This leads to ADC0DAT scaled by 2. 11 = not defined.
5, 4	Current channel ADC reference select. 00 = internal, 1.2 V precision reference selected. In ADC low power mode, the voltage reference selection is controlled by ADCMDE[5]. 01 = external reference inputs (VREF, GND_SW) selected. 10 = external reference inputs divided-by-2 (VREF, GND_SW)/2 selected, which allows an external reference up to REG_AVDD. 11 = (REG_AVDD, AGND) divided-by-2 selected.
3 to 0	Current channel ADC gain select. The nominal I-ADC full-scale input voltage = (VREF/gain). 0000 = I-ADC gain of 1. 0001 = I-ADC gain of 2. 0010 = I-ADC gain of 4. 0011 = I-ADC gain of 8. 0100 = I-ADC gain of 16. 0101 = I-ADC gain of 32. 0110 = I-ADC gain of 64. 0111 = I-ADC gain of 128. 1000 = I-ADC gain of 256. 1001 = I-ADC gain of 512. 1xxx = I-ADC gain is undefined.

Current Channel ADC Data Register

Name: ADC0DAT

Address: 0xFFFF0520

Default Value: 0x0000

Access: Read only

Function: This ADC data MMR holds the 16-bit conversion result from the I-ADC. The ADC does not update this MMR if the ADC0 conversion result ready bit (ADCSTA[0]) is set. A read of this MMR by the MCU clears all asserted ready flags (ADCSTA[2:0]).

Voltage Channel ADC Data Register

Name: ADC1DAT

Address: 0xFFFF0524

Default Value: 0x0000

Access: Read only

Function: This ADC data MMR holds the 16-bit voltage conversion result from the V-/T-ADC. The ADC does not update this MMR if the voltage conversion result ready bit (ADCSTA[1]) is set. If I-ADC is not active, a read of this MMR by the MCU clears all asserted ready flags (ADCSTA[2:1]).

Temperature Channel ADC Data Register

Name: ADC2DAT

Address: 0xFFFF0528

Default Value: 0x0000

Access: Read only

Function: This ADC data MMR holds the 16-bit temperature conversion result from the V-/T-ADC. The ADC does not update this MMR if the temperature conversion result ready bit (ADCSTA[2]) is set. If I-ADC and V-ADC are not active, a read of this MMR by the MCU clears all asserted ready flags (ADCSTA[2]). A read of this MMR clears ADCSTA[2].

Current Channel ADC Offset Calibration Register

Name: ADC0OF

Address: 0xFFFF0530

Default Value: Part specific, factory programmed

Access: Read/write

Function: This ADC offset MMR holds a 16-bit offset calibration coefficient for the I-ADC. The register is configured at power-on with a factory default value. However, this register automatically overwrites if an offset calibration of the I-ADC is initiated by the user via bits in the ADCMDE MMR. User code can write to this calibration register only if the ADC is in idle mode. An ADC must be enabled and in idle mode before being written to any offset or gain register. The ADC must be in idle mode for at least 23 μs.

Voltage Channel ADC Offset Calibration Register

Name: ADC1OF

Address: 0xFFFF0534

Default Value: Part specific, factory programmed

Access: Read/write

Function: This offset MMR holds a 16-bit offset calibration coefficient for the voltage channel. The register is configured at power-on with a factory default value. However, this register is automatically overwritten if an offset calibration of the voltage channel is initiated by the user via bits in the ADCMDE MMR. User code can write to this calibration register only if the ADC is in idle mode. An ADC must be enabled and in idle mode before being written to any offset or gain register. The ADC must be in idle mode for at least 23 μs.

Temperature Channel ADC Offset Calibration Register

Name: ADC2OF

Address: 0xFFFF0538

Default Value: Part specific, factory programmed

Access: Read/write

Function: This ADC offset MMR holds a 16-bit offset calibration coefficient for the temperature channel. The register is configured at power-on with a factory default value. However, this register is automatically overwritten if an offset calibration of the temperature channel is initiated by the user via bits in the ADCMDE MMR. User code can write to this calibration register only if the ADC is in idle mode. An ADC must be enabled and in idle mode before being written to any offset or gain register. The ADC must be in idle mode for at least 23 μs.

Current Channel ADC Gain Calibration Register

Name: ADC0GN

Address: 0xFFFF053C

Default Value: Part specific, factory programmed

Access: Read/write

Function: This gain MMR holds a 16-bit gain calibration coefficient for scaling the I-ADC conversion result. The register is configured at power-on with a factory default value. However, this register is automatically overwritten if a gain calibration of the I-ADC is initiated by the user via bits in the ADCMDE MMR. User code can write to this calibration register only if the ADC is in idle mode. An ADC must be enabled and in idle mode before being written to any offset or gain register. The ADC must be in idle mode for at least 23 μs.

TIMER4—STI TIMER

Timer4 is a general-purpose, 16-bit up/down counter timer with a programmable prescaler. Timer4 can be clocked from the core clock or from the low power 32.768 kHz oscillator with a prescaler of 1, 16, 256, or 32,768.

Timer4 has a capture register (T4CAP) that can be triggered by the initial assertion of a selected IRQ source. After the capture register is triggered, the current timer value is copied to T4CAP, and the timer continues running. This feature can be used to determine the assertion of an event with increased accuracy.

Timer4 can also be used to drive the serial test interface (STI) peripheral.

The Timer4 interface consists of five MMRs: T4LD, T4VAL, T4CAP, T4CLRI, and T4CON. T4LD, T4VAL, and T4CAP are 16-bit registers that hold 16-bit unsigned integers. T4VAL and T4CAP are read only. T4CLRI is an 8-bit register. Writing any value to this register clears the interrupt. T4CON is a configuration MMR and is described in Table 57.

Timer4 Load Register

Name: T4LD

Address: 0xFFFF0380

Default Value: 0x0000

Access: Read/write

Function: This 16-bit register holds the 16-bit value that is loaded into the counter.

Timer4 Clear Register

Name: T4CLRI

Address: 0xFFFF038C

Access: Write only

Function: This 8-bit, write only MMR is written (with any value) by user code to clear the interrupt.

Timer4 Value Register

Name: T4VAL

Address: 0xFFFF0384

Default Value: 0xFFFF

Access: Read only

Function: This 16-bit register holds the current value of Timer4.

Time4 Capture Register

Name: T4CAP

Address: 0xFFFF0390

Default Value: 0x0000

Access: Read only

Function: This 16-bit register holds the 32-bit value captured by an enabled IRQ event.

Timer4 Control Register

Name: T4CON

Address: 0xFFFF0388

Default Value: 0x00000000

Access: Read/write

Function: This 32-bit MMR configures the mode of operation of Timer4.

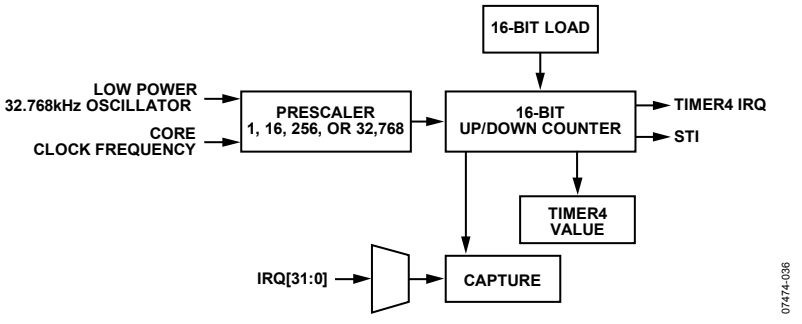

Figure 39. Timer4 Block Diagram

Table 58. External GPIO Pin to Internal Port Signal Assignments

Port	GPIO Pin	Port Signal	Functionality (Defined by GPxCON)
Port0	GPIO_0	P0.0	General-purpose I/O.
		IRQ0	External Interrupt Request 0.
		$\overline{SS}$	Slave select I/O for SPI.
	GPIO_1	P0.1	General-purpose I/O.
		SCLK	Serial clock I/O for SPI.
	GPIO_2	P0.2	General-purpose I/O.
		MISO	Master input, slave output for SPI.
	GPIO_3	P0.3	General-purpose I/O.
		MOSI	Master output, slave input for SPI.
	GPIO_4	P0.4	General-purpose I/O.
		ECLK	2.56 MHz clock output.
		P0.5[1]	High voltage serial interface.
		P0.6[1]	High voltage serial interface.
Port1	GPIO_5	P1.0	General-purpose I/O.
		IRQ1	External Interrupt Request 1.
		RxD	Pin for UART.
	GPIO_6	P1.1	General-purpose I/O.
		TxD	Pin for UART.
Port2	GPIO_7	Port 2.0	General-purpose I/O.
		IRQ4	External Interrupt Request 4.
		LIN output pin[2]	Used to read directly from LIN pin for conformance testing.
	GPIO_8	P2.1	General-purpose I/O.
		IRQ5	External Interrupt Request 5.
		LIN HV input pin[2]	Used to directly drive LIN pin for conformance testing.
	GPIO_11[2]	P2.4[2]	General-purpose I/O.
		LINRX[2]	LIN input pin.
	GPIO_12[2]	P2.5[2]	General-purpose I/O.
		LINTX[2]	LIN output pin.
	GPIO_13[1]	P2.6[1]	General-purpose I/O; STI data output.

[1] These signals are internal signals only and do not appear on an external pin. These pins are used along with HVCON as the 2-wire interface to the high voltage interface circuits.
[2] These pins/signals are internal signals only and do not appear on an external pin. The signals are used to provide the external pin diagnostic write (GPIO_12) and readback (GPIO_11) capability.

GPIO Port0 Set Register

Name: GP0SET

Address: 0xFFFF0D24

Access: Write only

Function: This 32-bit MMR allows user code to individually bit-address external GPIO pins to set them high only. User code can accomplish this using the GP0SET MMR without having to modify or maintain the status of the GPIO pins (as user code requires when using GP0DAT).

Table 65. GP0SET MMR Bit Designations

Bit	Description
31 to 21	Reserved. These bits are reserved and should be written as 0 by user code.
20	Port 0.4 set bit. Set to 1 by user code to set the external GPIO_4 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_4 pin.
19	Port 0.3 set bit. Set to 1 by user code to set the external GPIO_3 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_3 pin.
18	Port 0.2 set bit. Set to 1 by user code to set the external GPIO_2 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_2 pin.
17	Port 0.1 set bit. Set to 1 by user code to set the external GPIO_1 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_1 pin.
16	Port 0.0 set bit. Set to 1 by user code to set the external GPIO_0 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_0 pin.
15 to 0	Reserved. These bits are reserved and should be written as 0 by user code.

GPIO Port1 Set Register

Name: GP1SET

Address: 0xFFFF0D34

Access: Write only

Function: This 32-bit MMR allows user code to individually bit-address external GPIO pins to set them high only. User code can accomplish this using the GP1SET MMR without having to modify or maintain the status of the GPIO pins (as user code requires when using GP1DAT).

Table 66. GP1SET MMR Bit Designations

Bit	Description
31 to 18	Reserved. These bits are reserved and should be written as 0 by user code.
17	Port 1.1 set bit. Set to 1 by user code to set the external GPIO_6 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_6 pin.
16	Port 1.0 set bit. Set to 1 by user code to set the external GPIO_5 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_5 pin.
15 to 0	Reserved. These bits are reserved and should be written as 0 by user code.

GPIO Port2 Set Register

Name: GP2SET

Address: 0xFFFF0D44

Access: Write only

Function: This 32-bit MMR allows user code to individually bit-address external GPIO pins to set them high only. User code can accomplish this using the GP2SET MMR without having to modify or maintain the status of the GPIO pins (as user code requires when using GP2DAT).

Table 67. GP2SET MMR Bit Designations

Bit	Description
31 to 23	Reserved. These bits are reserved and should be written as 0 by user code.
22	Port 2.6 set bit. Set to 1 by user code to set the external GPIO_13 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_13 pin.
21	Port 2.5 set bit. Set to 1 by user code to set the external GPIO_12 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_12 pin.
20 to 18	Reserved. These bits are reserved and should be written as 0 by user code.
17	Port 2.1 set bit. Set to 1 by user code to set the external GPIO_8 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_8 pin.
16	Port 2.0 set bit. Set to 1 by user code to set the external GPIO_7 pin high. Clearing this bit to 0 via user software has no effect on the external GPIO_7 pin.
15 to 0	Reserved. These bits are reserved and should be written as 0 by user code.

GPIO Port0 Clear Register

Name: GP0CLR

Address: 0xFFFF0D28

Access: Write only

Function: This 32-bit MMR allows user code to individually bit-address external GPIO pins to clear them low only. User code can accomplish this using the GP0CLR MMR without having to modify or maintain the status of the GPIO pins (as user code requires when using GP0DAT).

Table 68. GP0CLR MMR Bit Designations

Bit	Description
31 to 21	Reserved. These bits are reserved and should be written as 0 by user code.
20	Port 0.4 clear bit. Set to 1 by user code to clear the external GPIO_4 pin low. Clearing this bit to 0 via user software has no effect on the external GPIO_4 pin.
19	Port 0.3 clear bit. Set to 1 by user code to clear the external GPIO_3 pin low. Clearing this bit to 0 via user software has no effect on the external GPIO_3 pin.
18	Port 0.2 clear bit. Set to 1 by user code to clear the external GPIO_2 pin low. Clearing this bit to 0 via user software has no effect on the external GPIO_2 pin.
17	Port 0.1 clear bit. Set to 1 by user code to clear the external GPIO_1 pin low. Clearing this bit to 0 via user software has no effect on the external GPIO_1 pin.
16	Port 0.0 clear bit. Set to 1 by user code to clear the external GPIO_0 pin low. Clearing this bit to 0 via user software has no effect on the external GPIO_0 pin.
15 to 0	Reserved. These bits are reserved and should be written as 0 by user code.

High Voltage Interface Control Register

Name: HVCON

Address: 0xFFFF0804

Default Value: Updated by kernel

Access: Read/write

Function: This 8-bit register acts as a command byte interpreter for the high voltage control interface. Bytes written to this register are interpreted as read or write commands to a set of four indirect registers related to the high voltage circuits. The HVDAT register is used to store data to be written to, or read back from, the indirect registers.

Table 71. HVCON MMR Write Bit Designations

Bit	Description
7 to 0	Command byte. Interpreted as 0x00 = read back High Voltage Register HVCFG0 into HVDAT. 0x01 = read back High Voltage Register HVCFG1 into HVDAT. 0x02 = read back High Voltage Status Register HVSTA into HVDAT. 0x03 = read back High Voltage Status Register HVMON into HVDAT. 0x08 = write the value in HVDAT to the High Voltage Register HVCFG0. 0x09 = write the value in HVDAT to the High Voltage Register HVCFG1.

Table 72. HVCON MMR Read Bit Designations

Bit	Description
7 to 3	Reserved.
2	Transmit command to high voltage die status. 1 = command completed successfully. 0 = command failed.
1	Read command from high voltage die status. 1 = command completed successfully. 0 = command failed.
0	Busy bit (read only). When user code reads this register, Bit 0 should be interpreted as the busy signal for the high voltage interface. This bit can be used to determine if a read request has completed. High voltage (read/write) commands as described in this table should not be written to HVCON unless busy = 0. Busy = 1, high voltage interface is busy and has not completed the previous command written to HVCON. Bit 1 and Bit 2 are not valid. Busy = 0, high voltage interface is not busy and has completed the command written to HVCON. Bit 1 and Bit 2 are valid.

UART SERIAL INTERFACE

The ADuC7036 features a 16,450-compatible UART. The UART is a full-duplex, universal, asynchronous receiver/transmitter. A UART performs serial-to-parallel conversion on data characters received from a peripheral device and performs parallel-to-serial conversion on data characters received from the ARM7TDMI. The UART features a fractional divider that facilitates high accuracy baud rate generation and a network addressable mode. The UART functionality is available on the GPIO_5/IRQ1/RxD and GPIO_6/ TxD pins of the ADuC7036.

The serial communication adopts an asynchronous protocol that supports various word lengths, stop bits, and parity generation options selectable in the configuration register.

BAUD RATE GENERATION

The ADuC7036 features two methods of generating the UART baud rate: normal 450 UART baud rate generation and ADuC7036 fractional divider baud rate generation.

Normal 450 UART Baud Rate Generation

The baud rate is a divided version of the core clock using the value in COMDIV0 and COMDIV1 MMRs (each is a 16-bit value, DL). The standard baud rate generator formula is

$$Baud\ Rate = \frac{20.48\ \text{MHz}}{2^{CD} \times 16 \times 2 \times DL} \quad (1)$$

Table 79 lists common baud rate values.

Table 79. Baud Rate Using the Standard Baud Rate Generator

Baud Rate (bps)	CD	DL	Actual Baud Rate	% Error
9600	0	0x43	9552	0.50%
19,200	0	0x21	19,394	1.01%
115,200	0	0x6	106,667	7.41%
9600	3	0x8	10,000	4.17%
19,200	3	0x4	20,000	4.17%
115,200	3	0x1	80,000	30.56%

Fractional Divider

The fractional divider, combined with the normal baud rate generator, allows the generation of accurate, high speed baud rates.

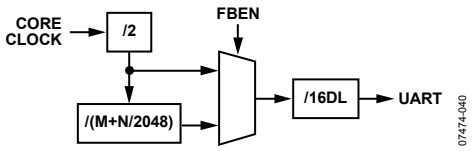

Figure 43. Fractional Divider Baud Rate Generation

Calculation of the baud rate using a fractional divider is as follows:

$$Baud\ Rate = \frac{20.48\ \text{MHz}}{2^{CD} \times 16 \times DL \times 2 \times (M + \frac{N}{2048})} \quad (2)$$

$$M + \frac{N}{2048} = \frac{20.48\ \text{MHz}}{Baud\ Rate \times 2^{CD} \times 16 \times DL \times 2}$$

where:
CD is the clock divider.
DL is the divisor latch.
M is the integer part of the divisor; a fractional divider divides an input by a nonwhole number, M.N.
N is the fractional part of the divisor; a fractional divider divides an input by a nonwhole number, M.N.

Table 80 lists common baud rate values.

Table 80. Baud Rate Using the Fractional Baud Rate Generator

Baud Rate (bps)	CD	DL	M	N	Actual Baud Rate	% Error
9600	0	0x42	1	21	9598.55	0.015%
19,200	0	0x21	1	21	19,197.09	0.015%
115,200	0	0x5	1	228	115,177.51	0.0195%

UART Interrupt Enable Register 0

Name: COMIEN0

Address: 0xFFFF0704

Default Value: 0x00

Access: Read/write

Function: This 8-bit register enables and disables the individual UART interrupt sources.

Table 84. COMIEN0 MMR Bit Designations

Bit	Name	Description
7 to 4		Reserved. Not used.
3	EDSSI	Reserved. This bit should be written as 0.
2	ELSI	RxD status interrupt enable bit. Set by the user to enable generation of an interrupt if any of the COMSTA0[3:1] register bits are set. Cleared by the user.
1	ETBEI	Enable transmit buffer empty interrupt. Set by the user to enable an interrupt when the buffer is empty during a transmission, that is, when COMSTA0[5] is set. Cleared by the user.
0	ERBFI	Enable receive buffer full interrupt. Set by the user to enable an interrupt when the buffer is full during a reception. Cleared by the user.

UART Interrupt Identification Register 0

Name: COMIID0

Address: 0xFFFF0708

Default Value: 0x01

Access: Read only

Function: This 8-bit register reflects the source of the UART interrupt.

Table 85. COMIID0 MMR Bit Designations

Bits[2:1] Status Bits	Bit 0 NINT	Priority	Definition	Clearing Operation
00	1		No interrupt	
11	0	1	Receive line status interrupt	Read COMSTA0
10	0	2	Receive buffer full interrupt	Read COMRX
01	0	3	Transmit buffer empty interrupt	Write data to COMTX or read COMIID0
00	0	4	Reserved	Reserved

LIN Hardware Synchronization Status Register

Name: LHSSTA

Address: 0xFFFF0780

Default Value: 0x00000000

Access: Read only

Function: This LHS status register is a 32-bit register whose bits reflect the current operating status of the LIN interface.

Table 92. LHSSTA MMR Bit Designations

Bit	Description
31 to 7	Reserved. These read only bits are reserved for future use.
6	Rising edge detected (BSD mode only). Set to 1 by hardware to indicate a rising edge has been detected on the BSD bus. Cleared to 0 after user code reads the LHSSTA MMR.
5	LHS reset complete flag. Set to 1 by hardware to indicate an LHS reset command has completed successfully. Cleared to 0 after user code reads the LHSSTA MMR.
4	Break field error. Set to 1 by hardware and generates an LHS interrupt (IRQEN[7]) when the 12-bit break timer (LHSVAL1) register overflows to indicate the LIN bus has stayed low too long, thus indicating a possible LIN bus error. Cleared to 0 after user code reads the LHSSTA MMR.
3	LHS compare interrupt. Set to 1 by hardware when the value in LHSVAL0 (LIN synchronization bit timer) equals the value in the LHSCMP register. Cleared to 0 after user code reads the LHSSTA MMR.
2	Stop condition interrupt. Set to 1 by hardware when a stop condition is detected. Cleared to 0 after user code reads LHSSTA MMR.
1	Start condition interrupt. Set to 1 by hardware when a start condition is detected. Cleared to 0 after user code reads LHSSTA MMR.
0	Break timer compare interrupt. Set to 1 by hardware when a valid LIN break condition is detected. A LIN break condition is generated when the LIN break timer value reaches the break timer compare value (see the LHSVAL1 in the LIN Hardware Synchronization Break Timer1 Register section for more information). Cleared to 0 after user code reads the LHSSTA MMR.

Example LIN Hardware Synchronization Routine

Using the following C-source code LIN initialization routine, LHSVAL1 begins to count on the first falling edge received on the LIN bus. If LHSVAL1 exceeds the value written to LHSVAL1, in this case 0x3F, a break compare interrupt is generated.

On the next falling edge, LHSVAL0 begins counting. LHSVAL0 monitors the number of falling edges and compares it to the value written to LHSCON1[7:4]. In this example, the number of edges to monitor is six falling edges of the LIN frame, or the five falling edges of the sync byte. When this number of falling edges is received, a stop condition interrupt is generated. It is at this point that the UART is configured to receive the protected identifier.

The UART must be gated through LHSCON0[8] before the LIN bus returns high. If the LIN bus returns high when UART is not gated, UART communication errors may occur. This process is shown in detail in Figure 52. Example code to ensure the success of this process follows Figure 49.

```
void LIN_INIT(void )
{
       char HVstatus;
       GP2CON = 0x110000;  // Enable LHS on GPIO pins

       LHSCON0 = 0x1;      // Reset LHS interface

       do{
              HVDAT = 0x02; // Enable normal LIN Tx mode
              HVCON = 0x08; // Write to Config0
              do{
                     HVstatus = HVCON;
              }
              while(HVstatus & 0x1); // Wait until command is finished
       }
       while (!(HVstatus & 0x4)); // Transmit command is correct

       while((LHSSTA & 0x20) == 0 )
       {                   // Wait until the LHS hardware is reset
       }

       LHSCON1 = 0x062;    // Sets stop edge as the fifth falling edge
                           // and the start edge as the first falling
                           // edge in the sync byte
       LHSCON0 = 0x0114;   // Gates UART Rx line, ensuring no interference
                           // from the LIN into the UART
                           // Selects the stop condition as a falling edge
                           // Enables generation of an interrupt on the
                           // stop condition
                           // Enables the interface
       LHSVAL1 = 0x03F;    // Sets number of 131 kHz periods to generate a break interrupt
                           // 0x3F / 131 kHz ~ 480 µs, which is just over 9.5 Tbits
```

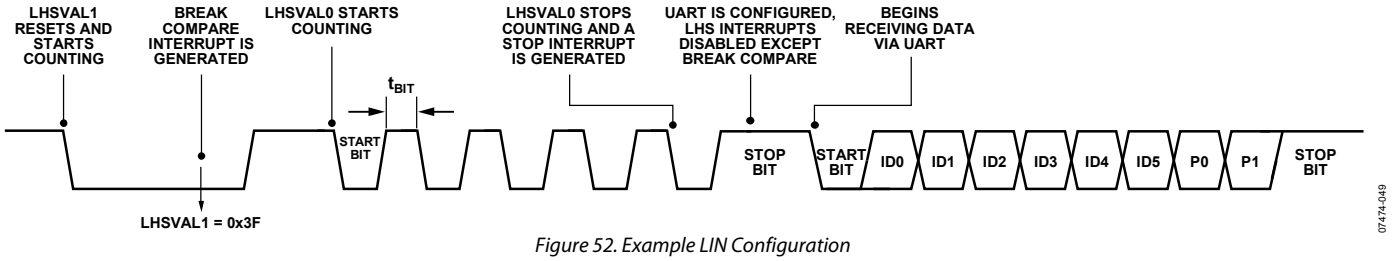

Figure 52. Example LIN Configuration

```
       while((GP2DAT & 0x10 ) == 0 )
       {}                  // Wait until LIN Bus returns high
       LHSCON0 = 0x4;      // Enable LHS to detect Break Condition Ungate RX Line
                           // Disable all Interrupts except Break Compare Interrupt
       IRQEN = 0x800;      // Enable UART Interrupt
                           // The UART is now configured and ready to be used for LIN
```

LIN Diagnostics

The ADuC7036 features the capability to nonintrusively monitor the current state of the LIN/BSD pin. This readback functionality is implemented using GPIO_11. The current state of the LIN/BSD pin is contained in GP2DAT[4].

It is also possible to drive the LIN/BSD pin high and low through user software, allowing the user to detect open-circuit conditions. This functionality is implemented via GPIO_12. To enable this functionality, GPIO_12 must be configured as a GPIO through GP2CON[20]. After it is configured, the LIN/BSD pin can be pulled high or low using GP2DAT.

The ADuC7036 also features short-circuit protection on the LIN/BSD pin. If a short-circuit condition is detected on the LIN/BSD pin, HVSTA[2] is set. This bit is cleared by reenabling the LIN driver using HVCFG1[3]. It is possible to disable this feature through HVCFG1[2].

LIN Operation During Thermal Shutdown

When a thermal event occurs, that is, when HVSTA[3] is set, LIN communications continue uninterrupted.

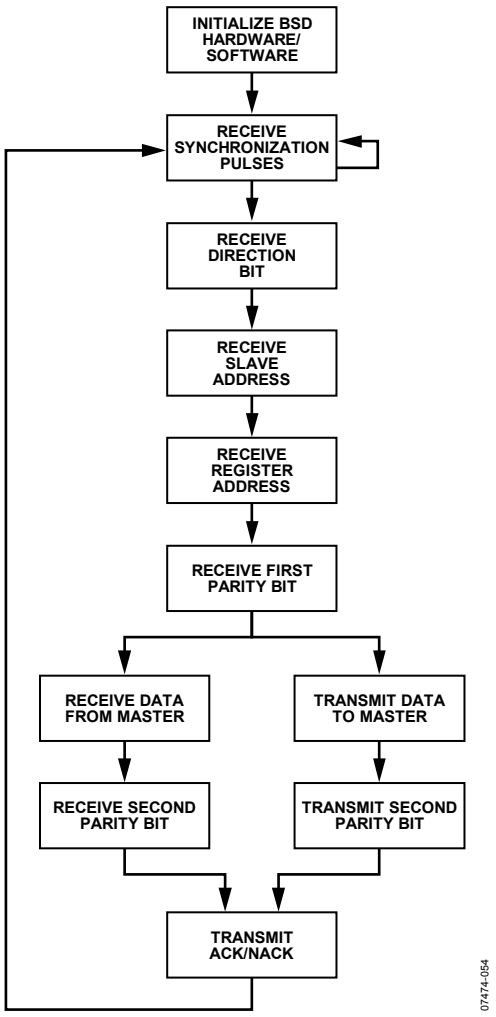

Figure 57. BSD Slave Node State Machine

BSD DATA RECEPTION

To receive data, the LIN/BSD peripheral must first be configured in BSD mode where LHSCON0[6] = 1. In this mode, LHSCON0[8] should be set to ensure that the LHS break timer (see the LIN Hardware Synchronization Break Timer1 Register section) generates an interrupt on the rising edge of the BSD bus.

The LHS break timer is cleared and starts counting on the falling edge of the BSD bus; the timer is subsequently stopped and generates an interrupt on the rising edge of the BSD bus. Given that the LHS break timer is clocked by the low power 131 kHz oscillator, the value in LHSVAL1 can be interpreted by user code to determine if the received data bit is a BSD sync pulse, 0, or 1.

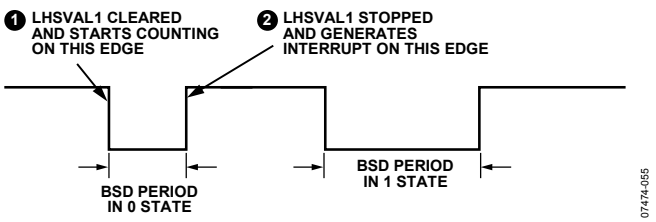

Figure 58. Master Transmit, Slave Read

BSD DATA TRANSMISSION

User code forces the GPIO_12 signal low for a specified time to transmit data in BSD mode. In addition, user code uses the sync timer (LHSVAL0), the LHS sync capture register (LHSCAP), and the LHS sync compare register (LHSCMP) to determine the length of time that the BSD bus should be held low for bit transmissions in the 0 or 1 state.

As described in the BSD Example Pulse Widths section, even when the slave is transmitting, the master always starts the bit transmission period by pulling the BSD bus low. If BSD mode is selected (LHSCON0[6] = 1), the LIN sync timer value is captured in LHSCAP on every falling edge of the BSD bus. The LIN sync timer runs continuously in BSD mode.

Then, user code can immediately force GPIO_12 low and read the captured timer value from LHSCAP. Next, the user can calculate how many clock periods (with a 5 MHz clock) should elapse before the GPIO_12 is driven high for a pulse width in the 0 or 1 state. The calcaulated number can be added to the LHSCAP value and written into the LHSCMP register. If LHSCON0[5] is set, the sync timer, which continues to count (being clocked by a 5 MHz clock), eventually equals the LHSCMP value and generates an LHS compare interrupt (LHSSTA[3]).

The response to this interrupt should be to force the GPIO_12 signal (and, therefore, the BSD bus) high. The software control of the GPIO_12 signal, along with the correct use of the LIN synchronization timers, ensures that valid pulse widths in the 0 and 1 states can be transmitted from the ADuC7036, as shown in Figure 59. Again, care must be taken if switching from BSD write mode to BSD read mode, as described in Table 93 (see the LHSCON0[8] bit.)

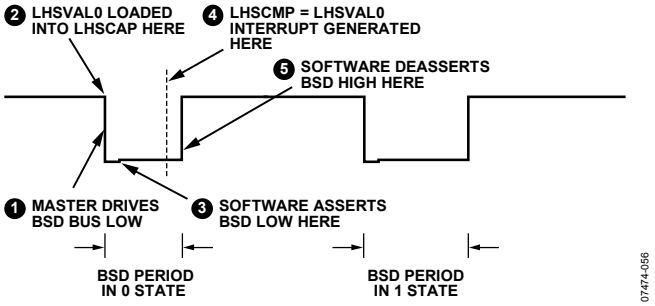

Figure 59. Master Read, Slave Transmit

WAKE-UP FROM BSD INTERFACE

The MCU core can be awakened from power-down via the BSD physical interface. Before entering power-down mode, user code should enable the start condition interrupt (LHSCON0[3]). When this interrupt is enabled, a high-to-low transition on the LIN/BSD pin generates an interrupt event and wakes up the MCU core.