AN_65XX_018 APRIL 2011

The SPI Interfaces of the 71M65xx

Introduction

Both the 71M653x and 71M654x families of electricity metering ICs offer SPI interfaces. These interfaces are described only briefly in the data sheets.

This Application note provides a more detailed description of the SPI interfaces in the 71M653x and 71M654x families of electricity metering ICs.

SPI General Description

SPI (serial peripheral interface bus) is a standard interface for digital communication on the board level with peripheral devices such as EEPROMs, ADCs and other devices. It implements a system consisting of one master and one or several slave devices. SPI uses four discrete signals or pins (the pin names used on the 71M653x are listed):

- PCSZ = chip select (sometimes referred to as “slave select”, in this case, “Z” is used to define a low-active chip select signal)
- PSCK = clock (sometimes referred to as SCK, serial clock)
- PSDI = master output/slave input (sometimes referred to as MOSI)
- PSDO = master input/slave output (sometimes referred to as MISO)

The clock, PSDI, and PCSZ signals are generated by the SPI master. The PSDO signal is generated by the slave device. In a multi-slave system, the slave is selected by an individual PCSZ signal.

SPI is generally a full-duplex interface, but individual implementations can be simplex, or half-duplex, depending on the type of device used for the slave.

Various operation modes are defined, that differ in the definition of the clock edge polarity and the type of clock edge used for PSDI and PSDO data. The SPI definition allows an almost completely free choice of message size, content, and purpose.

SPI Implementation for the 71M653x and 71M654x Families

The 71M653x and 71M654x families use a half-duplex protocol, i.e. the slave will never generate data while the master is emitting data. In terms of polarity, the 71M653x and 71M654x families use positive-going edges of the PSCK signal for data capture and an active-low PCSZ signal.

Slave data are clocked in and master data are clocked out (into the 71M653x) on rising edges of the clock. This operation mode is sometimes referred to as CPOL = 0 and CPHA = 0.

The SPI Interface of the 71M653x

General Description

SPI operations provide access to XRAM and to some, but not all, I/O RAM locations. The main purpose for this type of interfacing is to provide the host access to metrology data generated by the CE without having to involve the MPU of the 71M653x. This is particularly useful for configurations where the 71M653x acts as an FAE or metrology processor for the host.

The 71M653x provides arbitration between competing accesses to the XRAM data. This is particularly critical for accesses to CE RAM. CE RAM locations are routinely accessed by the CE, by the RTM, by the MPU, and by the SPI port. This arbitration precludes simultaneous accesses to the same addresses, but does not guarantee validity of the data. In other words, a datum accessed by the SPI may be overwritten during the next CE instruction cycle. I/O RAM accesses require special care (see below).

The code executing in the SPI host processor has to be written carefully in order to synchronize with the processes in the CE RAM.

Access speed for the 71M653x is up to 1 Mbit/s. The SPI port may operate at higher speeds under certain conditions. For SPI speeds higher than 1 Mbit/s, the following conditions apply:

- Write operations can be issued by the host at up to 2 Mbits/s.
- Read operations can be issued at up to 2 Mbit/s, if a minimum gap of 1 µs is inserted by the host between the last PSCK clock of the SPI address and the first clock of the data read. This gap will give the hardware of the 71M653x sufficient time to fetch and provide the read data.

Table 1 lists the subset of I/O RAM registers that is accessible via the SPI port.

In order to avoid access conflicts and to ensure data integrity, the following procedure must be used when accessing I/O RAM registers:

1) The SPI host should send a command with the command word 100x xxxx or 110x xxxx before the actual read or write command.
2) The SPI slave interface will load the command register and generate an INT2 interrupt upon receiving the command.
3) The MPU should service the interrupt and halt any external data memory operations to effectively grant the bus to the SPI.
4) When the SPI host finishes its I/O RAM access, it should send another command so the MPU can release the bus.

There are no issues with Data RAM access; SPI and the MPU will share the bus with no conflicts for Data RAM access.

Table 1: I/O RAM Registers Accessible via SPI

Name	Address (hex)	Bit Range	Read/Write
CE0	2000	7:3	RW
CE1	2001	7:0	RW
CE2	2002	5:3, 1:0	RW
CONFIG0	2004	7:6, 1:0	RW
CONFIG1	2005	5:2, 0	RW
VERSION	2006	7:0	R
CONFIG2	2007	7:0	RW
DIO0	2008	7, 4:0	RW
DIO1 to DIO6	2009 to 200E	6:4, 2:0	RW
–	200F	7:6, 3:2	RW
RTM0H	2060	1:0	RW
RTM0L	2061	7:0	RW
RTM1H	2062	1:0	RW
RTM1L	2063	7:0	RW
RTM2H	2064	1:0	RW
RTM2L	2065	7:0	RW
RTM3H	2066	1:0	RW
RTM3L	2067	7:0	RW
PLS_W	2080	7:0	RW
PLS_I	2081	7:0	RW
SLOT0 to SLOT9	2090 to 209A	7:0	RW

Name	Address (hex)	Bit Range	Read/Write
CE3	209D	3:0	RW
CE4	20A7	7:0	RW
CE5	20A8	7:0	RW
WAKE	20A9	7:5, 3:0	R
CONFIG3	20AC	7:0	RW
CONFIG4	20AD	7:0	RW
–	20AF	2:0	RW
SPI0	20B0	4, 0	RW
SPI1	20B1	4, 0	R
VERSION	20C8	7:0	R
CHIP_ID	20C9	7:0	R
TRIMSEL	20FD	4:0	RW
TRIMX	20FE	0	RW
TRIM	20FF	7:0	RW

SFR locations, i.e. the control registers internal to the 71M653x MPU, are not accessible via the SPI port. In cases where these registers have to be accessed, for example to control DIO pins, a protocol that uses the MPU has to be used for read and write operations involving the SFRs.

A typical SPI transaction is as follows: While PCSZ is high, the port is held in an initialized/reset state. During this state, PSDO is held in HiZ state and all transitions on PCLK and PSDI are ignored. When PCSZ falls, the port will begin the transaction on the first rising edge of PCLK. The transaction ends when PCSZ is raised. At this point, the SPI interrupt is generated. Some transactions may consist of a command only.

Typical read and write transactions are shown in Figure 1. The read transaction consists of the following parts:

1. 8-bit command word generated by the host
2. 16-bit address generated by the host
3. 8-bit datum provided by the slave (71M653x)
4. Optionally, more 8-bit data bytes (71M653x)

The write transaction consists of the following parts:

1. 8-bit command word generated by the host
2. 16-bit address generated by the host
3. 8-bit datum provided by the host
4. Optionally, more 8-bit data bytes provided by the host

The optional data bytes are part of an auto-increment mode, where the read or write address is incremented by 1 after every read or write operation and does not have to be generated by the host. This operation mode is useful for quickly accessing fields of adjacent data in one long SPI command sequence.

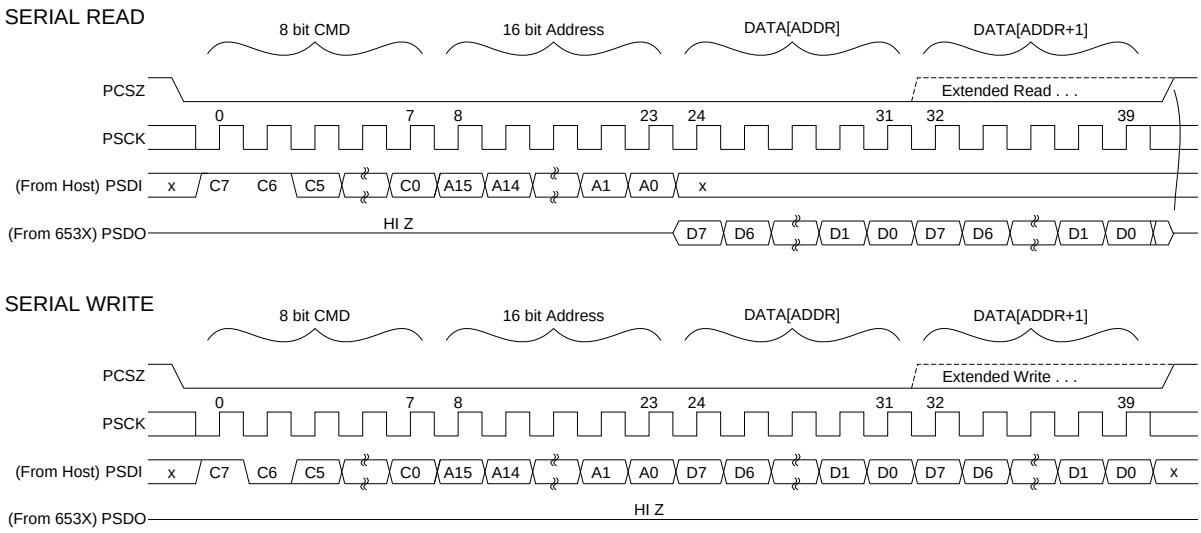


Figure 1: SPI Slave Port: Typical Read and Write operations

A read transaction performed at 2 Mbit/s is shown in Figure 2.

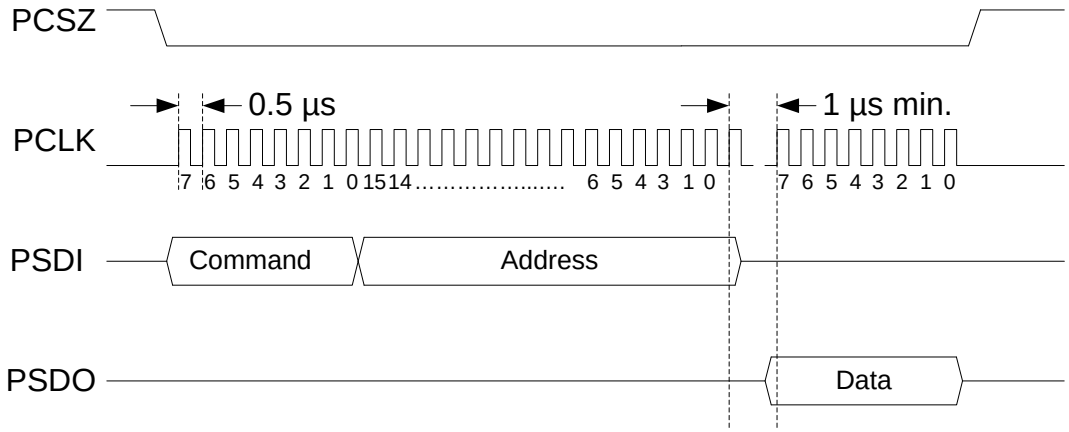


Figure 2: SPI Slave Port: Read Operation with Gap

Table 2 lists I/O RAM registers (bit fields) that are involved in SPI transactions.

Table 2: SPI Registers

Register Name	Description
SP_ADDR[15:8] *SP_ADDR[7:0]*	SPI Address. 16-bit address from the bus master. This register does not auto-increment and reading this register will not reflect the next available address after an auto-increment command.
SP_CMD	SPI command. 8-bit command from the bus master.
SPE	SPI port enable. Enables the SPI interface.
SPI_FLAG	SPI interrupt flag. The flag is set by the hardware and is cleared by the firmware writing a 0. Firmware using this interrupt should clear the spurious interrupt indication during initialization.

In order to allow access from the external host, the *SPE* bit has to be set. The *SP_CMD* and *SP_ADDR[15:0]* bit fields contain a copy of the command word and address sent by the SPI master.

The *SPI_FLAG* flag bit will be set upon every SPI transaction regardless of whether the command is 11xx xxxx or 10xx xxxx. The *SP_ADDR[15:0]* bit field is for writing purposes by the host only. Data read from *SP_ADDR[15:0]* will not contain the next available SPI address after an auto-increment operation.

Note: The data sheets up to revision 1.2 misstated the effect of the command word on interrupt generation.

Table 3 lists the command description as given in the data sheets for the 71M6531/6532 after revision 1.3 and for the 71M6533/6534 data sheets after revision 1.2 (future releases as of April 2011).

Table 3: SPI Command Description

Command	Description
11xx xxxx ADDR Byte0 ... ByteN	Read data starting at ADDR. The address value provided in ADDR will be automatically incremented until PCSZ is raised. Upon completion: *SP__CMD*=11xx xxxx An MPU interrupt is generated.
10xx xxxx ADDR Byte0 ... ByteN	Write data starting at ADDR. The address value provided in ADDR will be automatically incremented until PCSZ is raised. Upon completion: *SP_CMD*=10xx xxxx
0xxx xxxx ADDR Byte0 ... ByteN	Commands other than 1xxx xxxx are ignored, but an SPI interrupt is still generated when PCSZ goes high.

Working with the SPI Interface of the 71M653x

System Diagram

Figure 3 shows how the 71M653x and the host processor can be connected. The striped lines to DIO3 and DIO4 of the host show optional connections for synchronization and reset. These options are:

- The YPULSE output can signal sag warnings to the host. This eliminates the overhead that would occur if the host had to read the *CE_STATUS* register to detect sag warnings.
- The XPULSE output can signal zero crossing information to the host.
- The RESET pin, when connected to a DIO pin of the host, can be used by the host to force the 71M653x to a defined state.
- The TMUXOUT pin can be used to select either the XFER_BUSY, CE_BUSY, or MUX_SYNC signals to synchronize the host to the CE and ADC processes in the 71M653x.

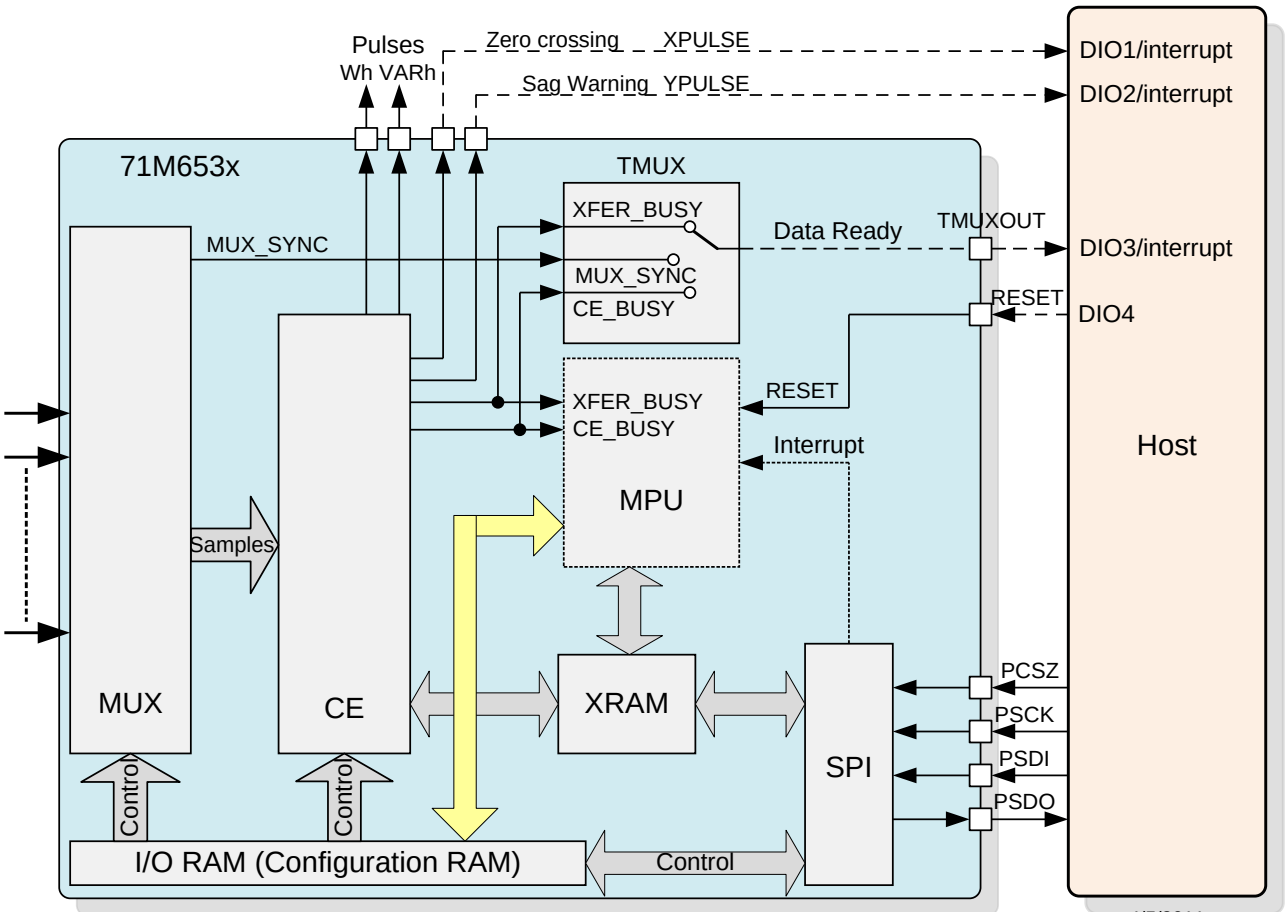
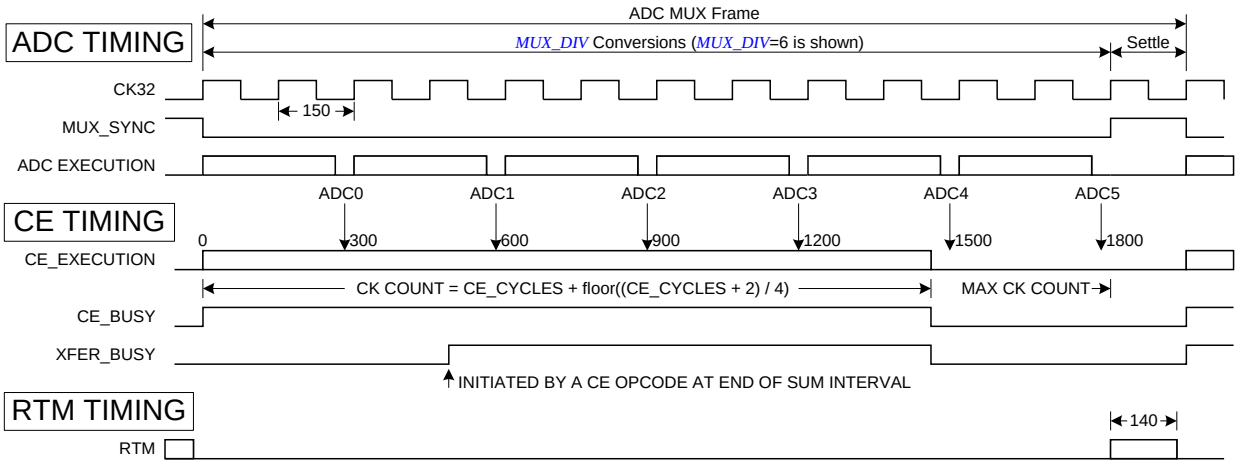


Figure 3: System Overview and Data Flow

Synchronizing Accesses with the CE Code Flow

Figure 4 was copied from the data sheet. It shows the CE activity in relation to the settings of the multiplexer in a poly-phase application. ADC data are generated after two CK32 cycles for each input, usually in the sequence IA, VA, IB, VB, IC, and VC. For this example, the entire ADC multiplexer frame lasts approximately 396 µs, which is equivalent to a sample frequency of 2520.6 Hz. Note that the CE_BUSY signal going low indicates the completion of a CE code run, but not necessarily the completion of all data collection (the ADC5 conversion occurs after CE_BUSY is low in this example). In order to ensure that all ADC data is fresh and that the CE is in HALT mode, the time period when MUX_SYNC is high should be used to access front-end data such as sampled raw data or even intermediate CE. This is also the time period during which the RTM is active.



NOTES:
1. ALL DIMENSIONS ARE 4.9152 MHz CK COUNTS.
2. XFER_BUSY OCCURS ONCE EVERY (PRESAMPS * SUM_CYCLES) CODE PASSES.

Figure 4: Timing Relationship between ADC MUX and Compute Engine

It can be useful to set the TMUXOUT pin of the 71M653x to select the CE_BUSY signal and make this signal accessible to the SPI host processor in order to synchronize the accesses of the SPI host processor to the ADC multiplexer frame.

If the SPI host processor is more interested in processed energy values, it can synchronize to the XFER_BUSY interrupt, which also can be made accessible via the TMUXOUT pin of the 71M653x. At the end of each accumulation interval, transfer variables such as *WSUM_X*, *VnSQSUM_X*, *InSQSUM_X*, *FREQU_X*, *TEMP_X* and so on are available and stable, which is signaled by XFER_BUSY going low.

MPU Program

For several reasons, it is necessary to have a small MPU program in the flash memory of the 71M653x, even when the host takes over all post-processing:

- The MPU has to be prevented from executing unrelated code. With the flash memory mostly empty, the MPU will execute 0xFF op-codes until it runs into the CE code image. Executing the CE code image could have undesired results, e.g., changes to core I/O RAM settings, and must therefore be avoided.
- The external host cannot access the SFRs of the MPU directly. However, SFR access is required for accessing the DIO pins. A small “driver” must exist to support SFR access, for example if the host needs to control the DIO pins.
- Access to I/O RAM locations by the external host need special precautions (see above).
- A small MPU program is useful to load the CE data image into the XRAM space, initialize core settings of the IC (*PRE_SAMPS*, *EQU*, *FIR_LEN*, *SUM_CYCLES* and the interrupt vector table, set the *CE_LCTN*[7:0] pointer, set the *SECURE* bit, enable the SPI port, start the CE and ADC, trigger the watchdog timer, and perform other routine tasks in order to offload the external host.
- It is useful to provide an incrementing counter for XFER_BUSY interrupts for the host. This allows the host to track the accumulation cycles, ensuring that no cycle is lost.

Interrupts

As stated before, an interrupt is generated for each completed SPI access. MPU codes based on two distinctive SPI operations (as described in the 71M6531/6532 and 71M6533/6534 data sheets), i.e. one with interrupt generation and one without interrupt generation, need to be modified so that they tolerate interrupt generation on any SPI access.

The suggested program flow is shown in Figure 6:

- If the interrupt stems from a regular read operation, no action should be required.
- If the interrupt stems from a special read operation, indicated by a command word 110x xxxx, the MPU executes the commands associated with it. Examples are SFR write operations that require the MPU to translate, or I/O RAM read operations.
- If the interrupt stems from a regular write operation, no action should be required.
- If the interrupt stems from a special write operation, indicated by a command word 100x xxxx, the MPU executes the commands associated with it. Examples are SFR write operations that require the MPU to translate, or I/O RAM write operations.

Tools

A variety of tools can be used to test the SPI port function. Maxim has successfully used the following:

- FT2232H (by FTDI, www.ftdichip.com) Mini-Module, used with Maxim FT2232H-to-SPI Adapter
- FT4232H (by FTDI, www.ftdichip.com) USB High-Speed Evaluation Module, used with Maxim FT4232H-to-SPI Adapter (see Figure 5).

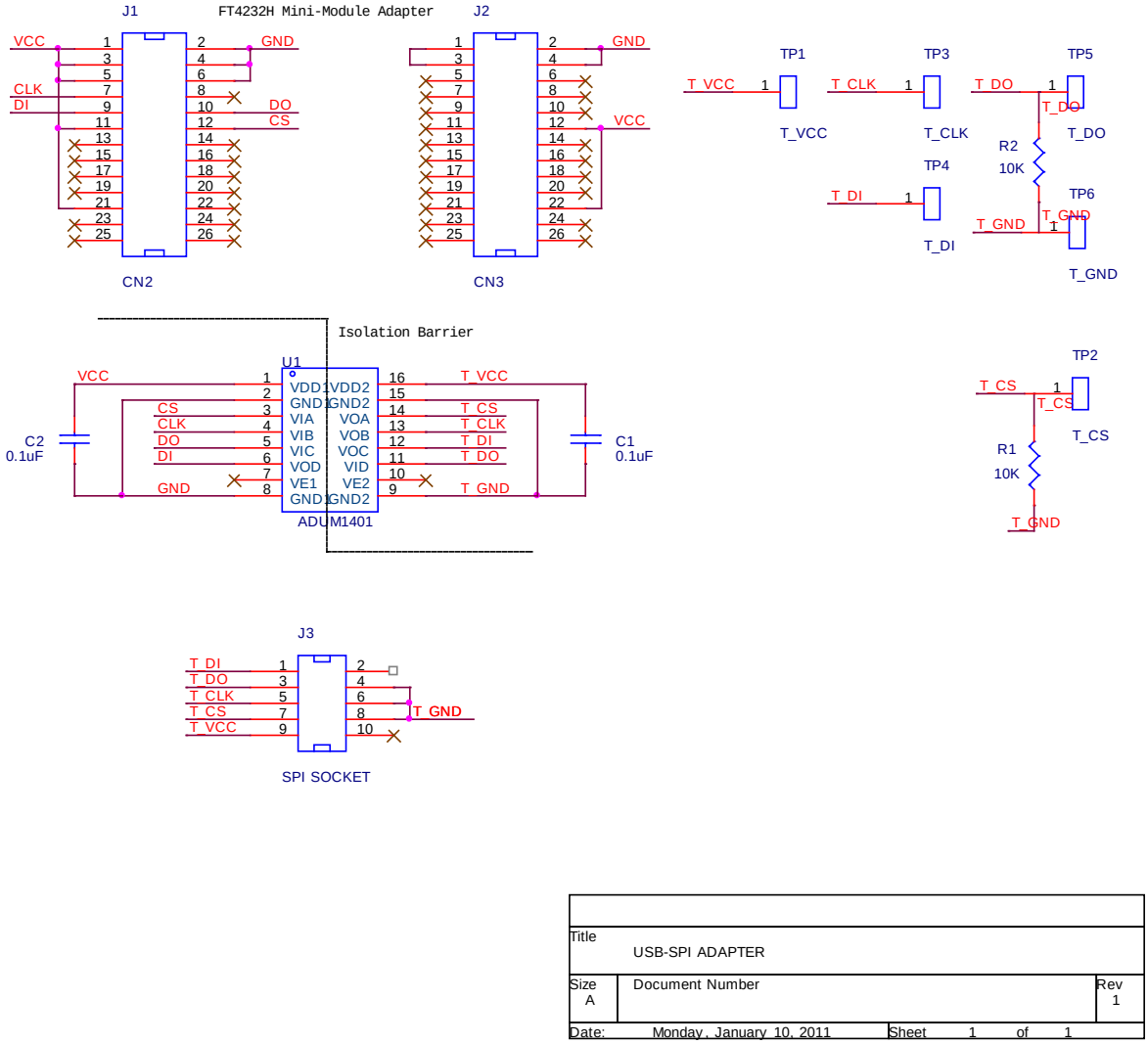


Figure 5: Board Adapter for FT4232H USB High-Speed Evaluation Module

Firmware to support cyclical testing based on Visual Basic can be provided on request.

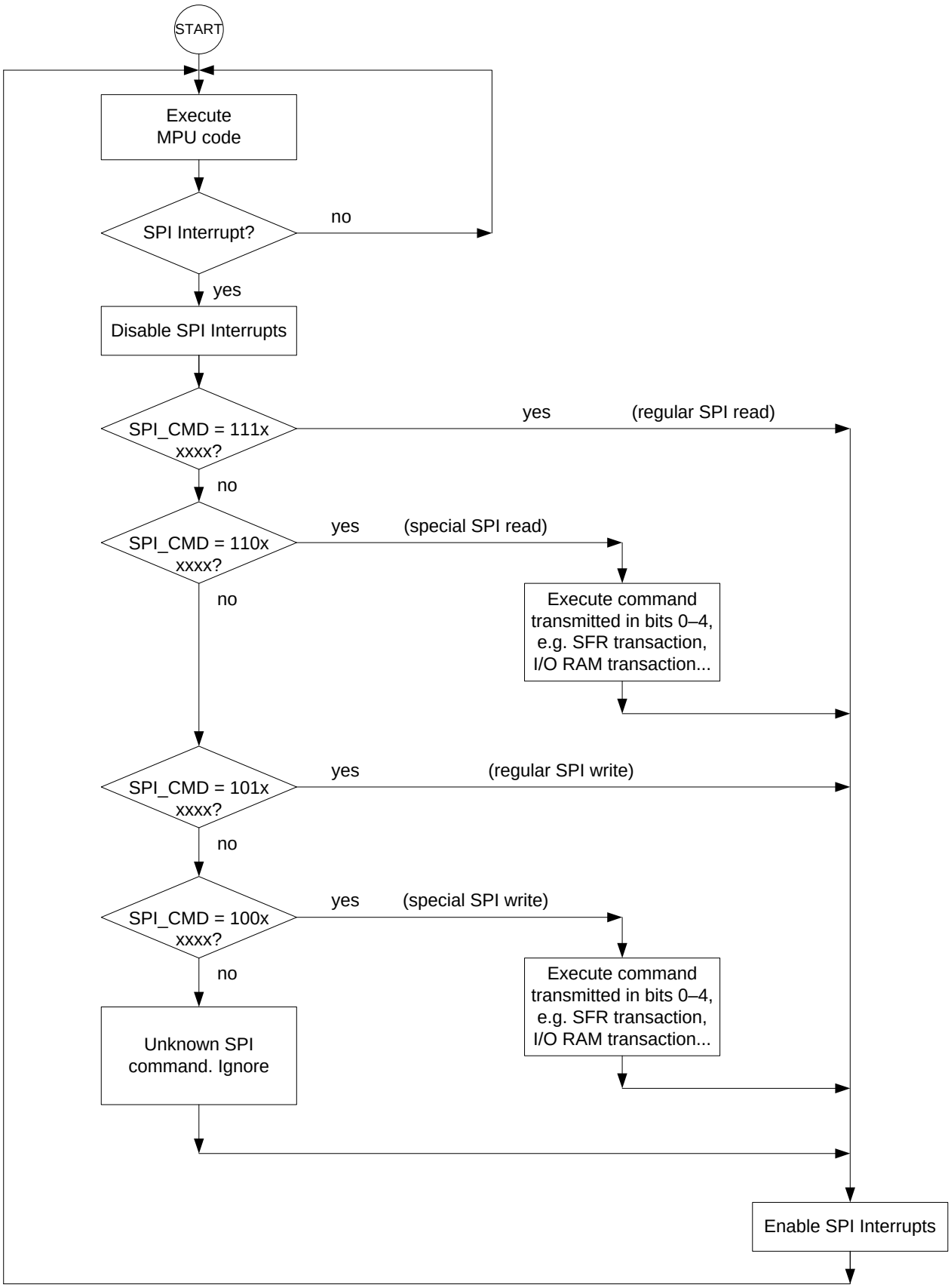


Figure 6: Interrupt Processing Flow Chart

The SPI Interface of the 71M654x

The description of the SPI interface in the 71M754x family of electricity metering ICs will be added at a later time.

Revision History

Revision	Date	Description
1.0	4/8/2011	First publication.
1.1	4/18/2011	DRAFT Fixed typos and inconsistent signal names. Added timing diagram for 2Mbit/s read transaction and schematics for FT4232 USB-SPI Adapter.

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600