Chapter 5

Ultra-Low Power Image Acquisition

This section focuses on a specific sub-system of the application circuit, with the goal to optimize the considered part as much as possible for low power consumption. Specifically, this section discusses the implementation of the camera sub-circuit of a multi-sensor smartwatch prototype. Together with acceleration, microphone and temperature, the image sensor is used for context recognition [4] and provide the end user with situation dependent smart features.

The decision to optimize the image sensor sub-circuit was made because the used image sensor has a non-standard interface, for which no hardware module is available on the microcontroller as it is for I2C or SPI. This example shall illustrate, how much energy can be saved by moving some functionality from software to the available hardware.

5.1. Related Work

Many contributions to build low power image sensors have been published in recent years. A comparison of different image sensors, their features and power consumption is given in [34]. All of them focus on low power consumption and have various special features that range from very high frame rate to reconfigurability as solar harvester. However, these approaches are research prototypes and not commercially available yet.

For a target application like a smartwatch it is of more interest, what image sensors and camera module solution are available commercially. A wide range of color camera modules is available with high frequency analog output signal. These modules feature good resolution and color images, but draw a power of 60 mW and more. [35] is one representative module for this category of cameras. Modules with an integrated circuit to compress the images draw even more power (hundreds of milliwatts) due to the additional

compression circuit. An example for these camera modules is the LinkSprite JPEG camera [36].

A digital output of the image data and integrated image processing, like black value and brightness adjustment is provided by sensors [37, 38]. These advanced processing features are combined with a high image acquisition speed of up to 100 images per second. Using these sensors, even if they have a power consumption comparable to the previously mentioned analog camera modules, might reduce the energy consumed for one image acquisition due to the faster acquisition.

An analog camera with a very low power consumption of only 2.2 mW is the Centeye Stonyman [39]. This low power consumption is achieved by directly applying the pixel's signal voltage to the analog output. A drawback of this sensor type is the required image post processing after the acquisition to reconstruct an image of good quality.

5.2. Sensor Hardware and Interface

The application circuit of the targeted smartwatch prototype for which the image sensor interfacing is optimized, is based on the ultra-low power microcontroller MSP430FR5969 [40, 26]. This microcontroller is optimized for ultra-low energy consumption and features 64 kilo-byte FRAM, a novel non-volatile memory technology with low energy consumption. The controller can run with up to 16MHz CPU clock and provides all feature typically found in an advanced microcontroller, like ADC, DMA, I2C, SPI, UART, PWM timer and many more. The MSP430 also provides different low power modes to optimized the energy consumption.

For the image acquisition on the target application a minimalistic Centeye Stonyman image sensor [39] was used. This chip uses only 5 digital inputs and one single analog output. Together with the two power supply connections, the chip uses only 8 lines to connect the sensor to a circuit, plus one optional enable input if multiple image sensors are used in the same circuit. The image sensor can directly be connected to a microcontroller with only a few additional components. To suppress noise in the supply voltage that is caused by clocked devices like microcontroller or communication modules, multiple bypass capacitors C_{bypass} with values of 1 nF, 100 nF and 10 μF are used.

The output impedance of the analog sensor output was decreased with a pull-down resistor R_{pd} at the output. This pull-down lowers the output voltage range of the sensor but results in better image quality when compared to a setup without. The exact value of this resistor, the suggested range is 5-20 kΩ, has no noticeable effect on the final image quality. For all the experiments shown later, the resistor value was fixed to $R_{pd} = 7.5$ kΩ. An overview of the image sensor sub-circuit that includes the mentioned components is depicted in Figure 5.1.

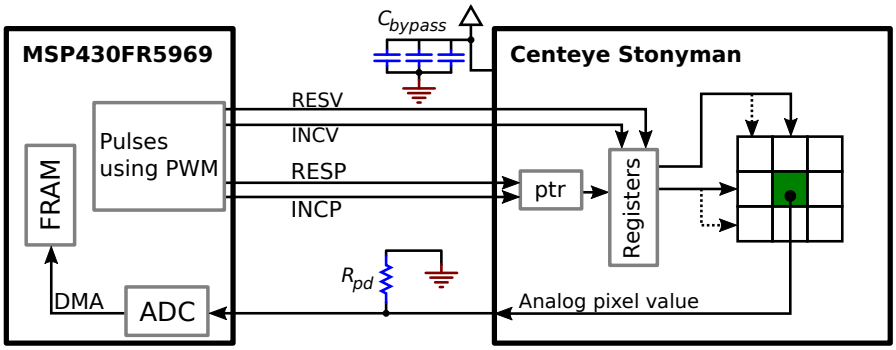

Figure 5.1.: Diagram shows the connection of the Centeye Stonyman image sensor to the microcontroller and a high level view of the used internal functionality.

The interaction with the image sensor is based on pulsed digital inputs and a single analog output for the pixel value. Four of the five input lines are used to reset or increment the register pointer (the increment pointer `INCP` and reset pointer `RESP` lines) and the value of the currently selected register (the increment value `INCV` and reset value `RESV` lines). A pulse on one of these lines changes the pointer or register value accordingly. The optional fifth pulse can be used to drive the integrated amplifier to scale the output voltage. This amplifier is recommended for low supply voltages < 4 V, like in the application targeted in this thesis. However, this functionality is not used in the final application, because extensive tests have shown that the amplifier introduces additional noise to the image. Using the low internal reference voltages of the MSP430 type used in the application circuit presented in this section, images of good quality can still be acquired.

With the column and row select registers, the MSP430 can select the pixel that is connected to the analog output. The output voltage of a pixel is a logarithmic function of the light intensity and allows using the sensor in a large range of light intensities. Using the integrated analog to digital converter (ADC) of the MSP430, the microcontroller can then convert this voltage to a pixel value and store it in the non-volatile FRAM memory. To acquire a complete image, every single one of the 112×112 pixels needs to be selected and converted.

The use of this very basic image sensor that directly outputs the raw pixel voltage requires some additional effort to reconstruct an image. The image sensor exhibits some sensor noise, so-called fixed pattern noise (FPN), that depends on the voltage supply, the light intensity and the configuration of the sensor. To remove this FPN from the captured image, a second image with uniform illumination needs to be acquired. For that, the camera is covered with white. This second image $\mathbf{I}_{\text{mask}}$, acquired with uniform illumination,can then be used as FPN mask (with an additional constant offset for the uniformly illuminated image). After subtracting this mask from the previously acquired raw image $\mathbf{I}_{\text{raw}}$ the reconstructed image can be extracted:

$$\mathbf{I}_{\text{reconstructed}} = \mathbf{I}_{\text{raw}} - \mathbf{I}_{\text{mask}}$$

(a) The raw image $\mathbf{I}_{\text{raw}}$.

(b) The mask $\mathbf{I}_{\text{mask}}$ for uniform sensor illumination.

(c) The reconstructed image $\mathbf{I}_{\text{reconstructed}}$.

Figure 5.2.: The reconstruction of the image $\mathbf{I}_{\text{reconstructed}}$ using the raw image $\mathbf{I}_{\text{raw}}$ and the mask $\mathbf{I}_{\text{mask}}$.

A sample of raw image data, the corresponding uniformly illuminated mask and the extracted image are shown in Figure 5.2. Taking an additional mask for each single image is impractical. Therefore, a bank of previously acquired masks for different luminosities can be used instead. Using the average image value or a single sample of the image sensor when configured as one huge pixel (see binning methods in [39]), the best mask for the given light intensity can be selected as FPN mask. A discussion and power analysis of the image acquisition and reconstruction using precalculated FPN masks can be found in [41].

5.3. Software Overview

After the discussion of the hardware and the basic interface in the previous section, this section focuses on the low-power implementation of the image sensor interface for the MSP430FR5969 ultra-low power microcontroller.

The Centeye Stonyman image sensor features 8 registers in total to configure the sensor. For the low-power application presented here, only 3 registers are of interest. The others are for features that were not used, which includes the mentioned amplifier or the pixel binning, or sensor bias values that need to be set only once at the beginning and the default values from [39] are already optimized [41, Section 3.1.1].

These register values all need to be set only once at power up of the sensor. For flexibility and because this needs to be done only once, this initial configuration is handled in software without the use of special hardware support.

The lower two registers at addresses `0x00` and `0x01` are used to select the columns and row of the pixel that is connected to the analog output. To acquire a complete image, the MSP430 must go once trough all pixels and sample the individual output signals with the ADC. Because there are 12544 pixels for the 112×112 pixel image sensor to sample,

acquiring an image takes some time and a lot of pulses need to be generated to switch trough the pixels. Because the pattern to select the images repeats itself very often and the GPIO toggling in software has a big runtime and energy overhead, the acquisition speed was increased and the power consumption reduced by using the hardware support of the MSP430 peripherals to generate the required pulses and automate ADC sampling of the pixel values. The combination of faster acquisition at lower power consumption leads to highly reduced energy consumption per image. The evaluation of how much energy can be saved is discussed in Section 5.4.

5.3.1. Low Power Image Acquisition Using PWM and DMA

For the low power implementation of the image acquisition, multiple timers in PWM mode are used to generate the recurring pulses for the pixel selection. These are started in sync with the ADC that is continuously sampling the analog output of the image sensor. The clock pulses to change the selected pixel are aligned such that the next pixel output is ready at the start of the next ADC sampling period. The DMA is then used to transfer the converted ADC values to the FRAM memory, when an ADC conversion has completed. Because the entire image acquisition can run in hardware, no CPU interaction is needed during the acquisition. This allows the CPU to be deactivated. It is woken up again when the complete image has been copied to the memory using a DMA interrupt.

An overview of the PWM pulse generation and the alignment of them with the ADC sample and conversion phases is depicted in Figure 5.3. The shortest sampling duration of the ADC lasts 4 cycles, followed by an additional clock cycle to synchronize with the conversion logic. The ADC conversion itself lasts another 14 clock cycles for the maximum output resolution of 12 bits that was chosen to get images of good quality. In total, this results in a new ADC conversion every 19 clock cycles. The ADC uses a sample and hold logic, which means that the input signal is only connected to the ADC input buffer during the sample phase. After the sample phase, changes at the ADC input does not affect the conversion result anymore and the sensor can already be switched to the next pixel. Because the camera output needs a minimal time of 2 μs after switching to a new pixel to stabilize its output voltage, it is important to switch the pixel as early as possible. Therefore, the PWM pulses follow directly after the synchronization cycle, which allows the stabilization of the sensor output voltage before the end of the next sample phase.

The pulse sequence for switching a pixel depends on the currently selected pixel. If only the next column of the same row should be selected, later referred to as column switch, there is only an `INCV` pulse to increment the column register, because the register pointer already points on register `0x00`. When the column register reached its highest value of 111, a more complex row switch is needed. This includes a reset of the column register value to 0 and an increment on the row register value. The additional pulses for the row switch are aligned around the `INCV` pulse that is already used in all ADC cycles to switch to the next column. First, the column register is reset to 0 using a `RESV` pulse,

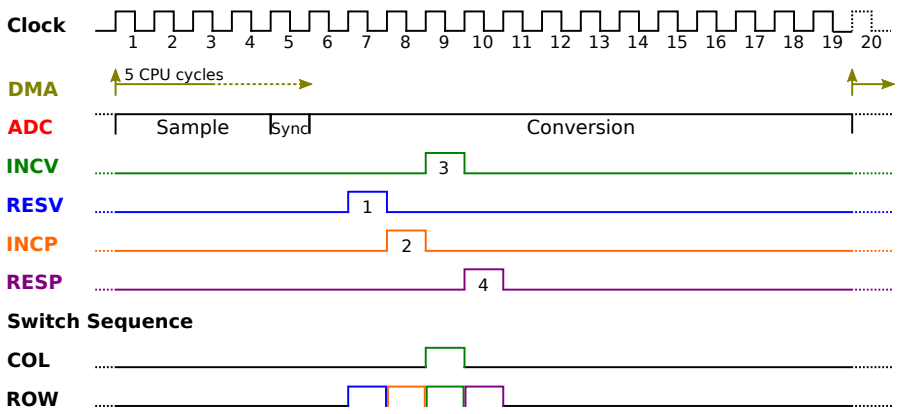

Figure 5.3.: Timing diagram of the different PWM signals and the ADC sampling and conversion times. A complete pixel acquisition needs 19 clock cycles for 12 bit ADC resolution.

followed by an `INCP` pulse to move the pointer to the row register at address `0x01`. After this pulse a `INCV` pulse is used to increment the row register value and shift the pixel selection to the next row. Finally, a `RESP` pulse makes sure that the pointer is set again to the column register at address `0x00`, such that the column switches in the subsequent periods only needs to pulse the `INCV` line.

For this pulse sequence 3 PWM timers are used: two of them have a period of 112×19 clock cycles to generate the three additional pulses for the row switch. The third timer is dedicated to the `INCV` line and has a period of 19 clock cycles to increment the column or row register every ADC conversion cycle. The PWM is configured to generate a pulse of exactly one clock cycle at the beginning of the period, except the `RESV` pulse, which pulses at the last clock cycle of the timer to share a common timer with the `INCP` pulse. To guarantee the depicted pulse sequence, the timer offsets need to be carefully chosen upon acquisition start. For the detailed timer offset values, please refer to the extended timing diagram in Appendix-E and the corresponding code and header files.

5.3.2. Timer Synchronization

One problem that cannot be solved directly in hardware is the synchronized start of the PWM timers, because there is no clock synchronization support on the MSP430. To solve this issue, an additional timer start offset was introduced for the two timer that are started after the other. The value of this offset needs to be evaluated for the target clock speed configuration by checking the shift of the pulses with an oscilloscope when this additional offsets are zero. This solution has the drawback that a for each clock configuration these offsets can be different and need to be checked and adjusted if necessary.

An alternative method could use the input capture capability of the timers: all three timer input trigger pins could be connected to a fourth GPIO pin that can then be used for triggering the start of all three timers simultaneously. However, this solution would need four of the GPIO pins which are already rare in the targeted application circuit, into which the camera should be integrated. Additionally, the start of the timers still needs to be synchronized with the start of the ADC sampling. The ADC has a start delay, because it has to wait for the internal reference voltage to stabilize before continuous sample and conversion starts. An interrupt flag indicates when the first analog sampling starts and needs to be polled until the ADC is ready, before the timers can be started.

5.3.3. ADC-to-DMA Trigger

In the low power implementation the DMA takes care of transferring new ADC conversion values from the ADC output register to the FRAM memory. For this purpose, the DMA can be configured to use the finished conversion flag of the ADC as trigger for a new transfer. During the implementation of this setup the MSP430 showed unexpected transfers even if no new data were available. Extensive debugging has shown that the ADC-DMA trigger is not being reset, even if the DMA transfer was executed properly. A discussion of this problem in the TI forum revealed that this problem can also be reproduced on other MSP430 microcontroller families and that this must be an erratum of the MSP430FR5069 that was not documented so far. The only workaround for this problem is to use a different DMA transfer mode and to imitate the behavior of the repeated single transfer mode with additional polling and manual reconfiguration at the end of an image acquisition.

5.4. Power Evaluation

The goal of the low power implementation of the image acquisition was to reduce the image acquisition power as far as possible. However, even more important is to reduce the total amount of energy spent for the acquisition of an image. To reduce the consumed energy, it might even make sense to run at a higher power consumption, but for a shorter amount of time. The main parameter that influences the power consumption is the clock frequency of the microcontroller. Lowering this frequency reduces the power consumption of the system, but increases the runtime of the program. On the other hand, an increased clock speed reduces the execution time, but at the cost of a higher power consumption. It is therefore inevitable to evaluate the proposed acquisition implementation for different clock frequencies to see the trade-off of energy consumption vs. clock frequency.

For the evaluation, two different implementations are compared for multiple frequencies. This allows the quantification of the energy saved with the low power implementation and the evaluation of which clock configuration minimizes the total energy spent for one image acquisition. These power measurements were carried out using an MSP430FR5969

Table 5.1.: Power consumption of the low power camera system consisting of an MSP430FR5969 and a Centeye Stonyman image sensor.

Acquisition Method	**CPU clock [MHz]**	**ADC clock [MHz]**	**Average Power consumption [μW]**	**Acquisition duration [ms]**	**Energy per image [μJ]**
Low Power	10.5	5.25	4928.64	46.666	230.0
Low Power	8	4	4313.40	60.996	263.1
Low Power	6	3	4391.72	81.130	356.3
Low Power	4	2	3777.38	122.307	462.0
Software	10.5	5.25	6075.63	468.001	2843.4
Software	8	4	4343.95	700.607	3043.4
Software	6	3	4776.37	735.140	3511.3
Software	4	2	3796.86	1047.207	3976.1
CPU sleep, sensor standby	off	off	0.87-1.05	-	-

LaunchPad [42]. The image sensor was connected to that board as described in Section 5.2 with special attention that the pulse lines are connected to the corresponding PWM output pin.

For the energy and timing measurements of the camera system, the EnergyTrace functionality of the LaunchPad was used [43]. With the integrated development environment *Code Composer Studio* [44], the energy trace was captured when running the test setup [1]. It is important to run the circuit without debugging, because this affects both the timing and the power consumption. The extracted time and energy measurements are then used to calculate the average power for each acquisition setup and the low power sleep phase between the acquisitions.

The power measurements for the software and optimized image acquisition method are shown in Table 5.1 for clock speeds between 2 and 5.25 MHz for the ADC and PWM and a CPU clock of always twice the ADC clock speed. The power consumption results show that for each clock configuration, the optimized method has a lower power consumption than the software implementation. However, the power savings are not always that significant, because only the CPU power can be reduced and the power hungry memory accesses still need to be performed with the low power method. As expected, the consumed power increases with higher clock speeds, except for the 6 MHz CPU clock, because a higher oscillator frequency needs to be generated and divided to get this CPU frequency. As reference, the last row lists the power consumption of the test setup when both the image sensor and the microcontroller are in standby mode, which is as low as 1 μW.

[1] The EnergyTrace tool measures the total amount of energy consumed by the circuit. For power analysis these energy measurements should be used to calculate the power. The internal derivation of the power trace seems not to work properly.

A huge improvement is observed for the image acquisition duration. With the low power image acquisition method the acquisition could be reduced down to 46.7 milliseconds, which results in a reduction of 90% when comparing the two fastest acquisitions of both methods. Including also the reduced power consumption, the energy per image acquisition was reduced by 92% from 2843 μJ down to 230 μJ with the presented low power implementation compared to the software solution. The analysis of the energy consumption also shows that the increased image acquisition speed outweighs the increased power consumption for higher clock speeds and that the most energy efficient image acquisition is reached for the highest ADC clock speed of 5.25 MHz.

Appendix E

Camera Software Reference

E.1. Function Reference

For the individual camera function, please check the doxygen documentation of the camera source code. To build this documentation the following two programs are required:

- Doxygen, download at `http://www.stack.nl/~dimitri/doxygen/`,
- Graphviz, download at `http://www.graphviz.org/`.

To build the documentation just run the following command in the project directory:

```
doxygen
```

or by using the eclipse plug-in "eclox" from `http://home.gna.org/eclox/#download` to build the documentation directly inside the Code Composer Studio.

The documentation can then be found in the `doc/` directory of the project. Open the `index.html` inside the `html/` subdir to start.

To setup and use the camera code, the following minimal sequence of function calls is needed:

1. `Camera_init()` to initialize the used GPIO pins and the PWM timers used for image acquisition.
2. `Camera_configure()` to set all the configuration registers of the image sensor to the desired values.
3. `Camera_setupADC()` to configure the ADC for the image acquisition. This function is useful if the ADC is shared with other analog inputs and needs to be called once at the beginning and after each other ADC conversion of other analog inputs.

4. `Camera_startAcquisition()` to start the low power image acquisition using PWM and DMA. After this function the processor can enter LPM0 (but not LPM1 or more, because access to the FRAM is still needed) and wait for the DMA interrupt that is fired once the image acquisition is complete.

E.2. Timing Header Definitions

For easy understanding of the PWM and ADC timing defines in the camera.h header files, we show again the PWM and timing diagram discussed in Section 5.3 in Figure E.1, including all defines and the corresponding time span for which they stand for.

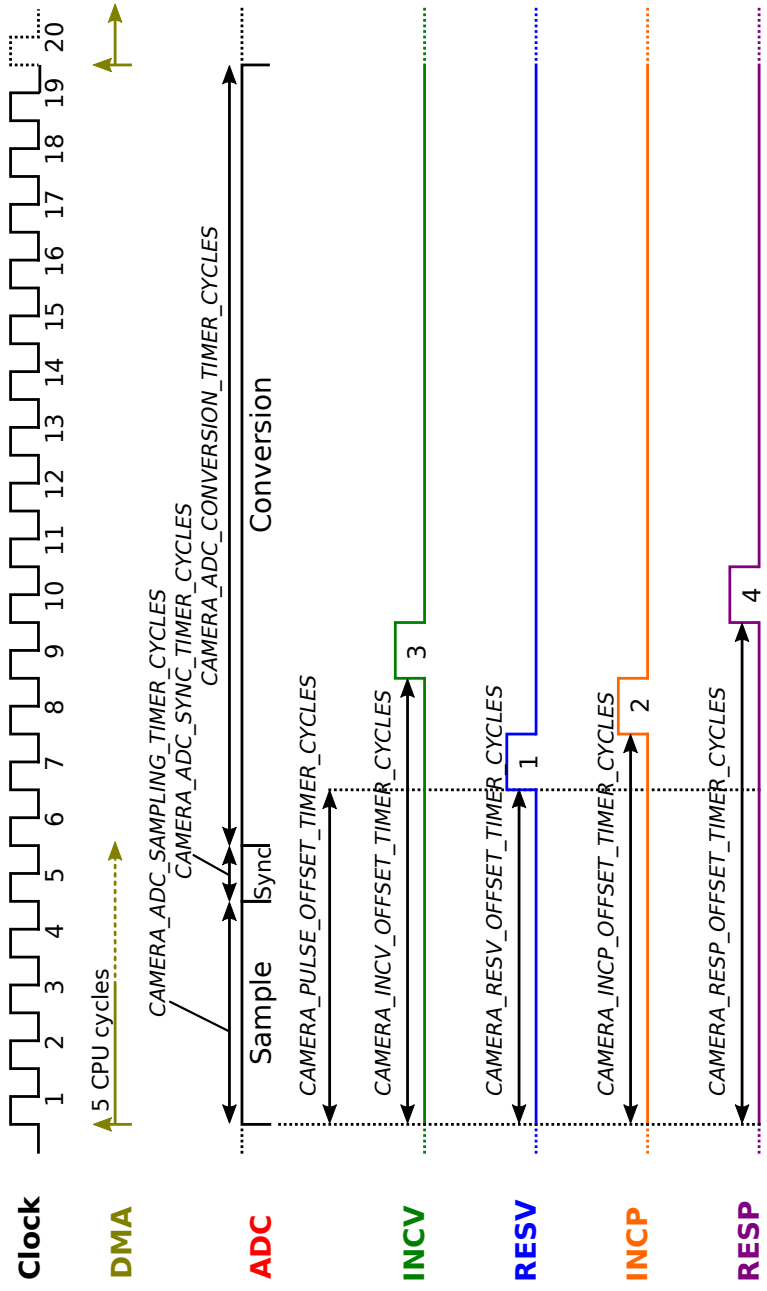

Figure E.1.: Timing diagram of the different PWM signals and the ADC sampling and conversion times, including the different camera header file defines.