Design Specification

Ville Grandin

Version 1.0

Status

Examined	-	
Approved		

Project identity

Spring 2006

Department of Electrical Engineering
Linköping Institute of Technology

Name	Responsibility	Telephone	Email
Mikael Lord	Documentation	070 – 400 89 77	miklo919@student.liu.se
Anna Lindefelt	Test	073 – 623 27 22	annli858@student.liu.se
Mikael Johansson	Customer	070 – 207 29 90	mikjo941@student.liu.se
Ville Grandin	Design	070 – 150 11 59	vilgr522@student.liu.se
Anders Jonasson	Implementation	073 – 694 30 76	andjo752@student.liu.se
Chistian Lyzell	Project leader	073 – 182 04 21	chrly059@student.liu.se

Homepage: http://www.cyd.liu.se/users/~andjo752/

Customer: DST Control
Customer contact: Jan-Erik Strömberg, 013–211080, janerik@dst.se
Course leader: Anders Hansson, 013–281681, hansson@isy.liu.se
Supervisor: Jeroen Hol, 013–282803, hol@isy.liu.se
Tutor: Janne Harju, 013–282804, harju@isy.liu.se

Contents

List of Figures

Document history

Version	Date	Changes	Sign	Reviewed
0.1	060317	First draw	ALL	AJ,MJ
1.0	060401	First draw	ALL	AJ,MJ

1 Introduction

The *General Avation* (GA) aircraft fleet has become very old (40+ year old aircrafts are now very common) thanks to reliable air frames and rapidly increasing prices of new light aircraft. A vast majority of these aircrafts are equipped with old-fashioned mechanical flight instruments, now reaching the end of their lifetime as the air frames become older. Instead of performing a complete upgrade of the entire instrument panel, which can result in costs often exceeding the value of the entire aircraft, one can complete the system with a much more affordable Micro EFIS. A Micro EFIS is, as the name implies, a small EFIS (*Electronic Flight Information System*), targeted for the GA market. It is intended as a backup system for the mechanical flight instruments.

1.1 Parties

Customer: Jan-Erik Strömberg, DST Control, Linköping

Supervisor: Jeroen Hol, Department of Electrical Engineering, Linköping Institute of Technology

Producent: A group of six students at Linköping Institute of Technology

1.2 Goal

The goal of this project is to design a stripped down version of a Micro EFIS and to evaluate the implementation based on state-of-the art *organic graphical display*; a so called OLED display. This to ensure high endurance at extreme temperatures (down to -30° C). The system shall be implemented on one single FPGA to meet the need to be *small in size* and have a *low cost*. Since it is intented as a backup system it must have a *high reliability*, be *independent of external systems* and *simple to use*. To increase the ease of upgrading the system and to *incrementally add functions*, it shall be built in modules.

1.3 Use

The system is intented as a backup for the mechanical flight instruments and the primary use is in the case of instrument failure.

1.4 Notation and abbreviations

The following abbreviations and notations will be used:

ADI Attitude and Direction Indicator

A/C Aircraft

DST DST Control AB: the project owner

EFIS Electronic Flight Information System: a standardised terminology used for a system of sensors, computers and displays designed to present critical flight data to the pilot

ESI Engine Status Indicator

GA General Aviation: non-commercial air traffic (including aircraft operated by companies for internal use only)

GND Ground Level

GUI Graphical User Interface

HSI Horizontal Situation Indicator

MSL Mean Sea Level

SPI Serial Peripheral Interface

2 System overview

2.1 Product components

The Micro EFIS consists of an instrument casing with specific measures ($60 \times 60 \times 50$ mm), with a user interface on a single monochrome display (128×64 pixels) and one single rotary mode selector switch. Inside the box there will be an FPGA (field programmable gate array), an RTC (real-time clock), and a FLASH memory. Externally the box will be connected to a power supply and sensor units. A user manual and technical documentation will be included at delivery.

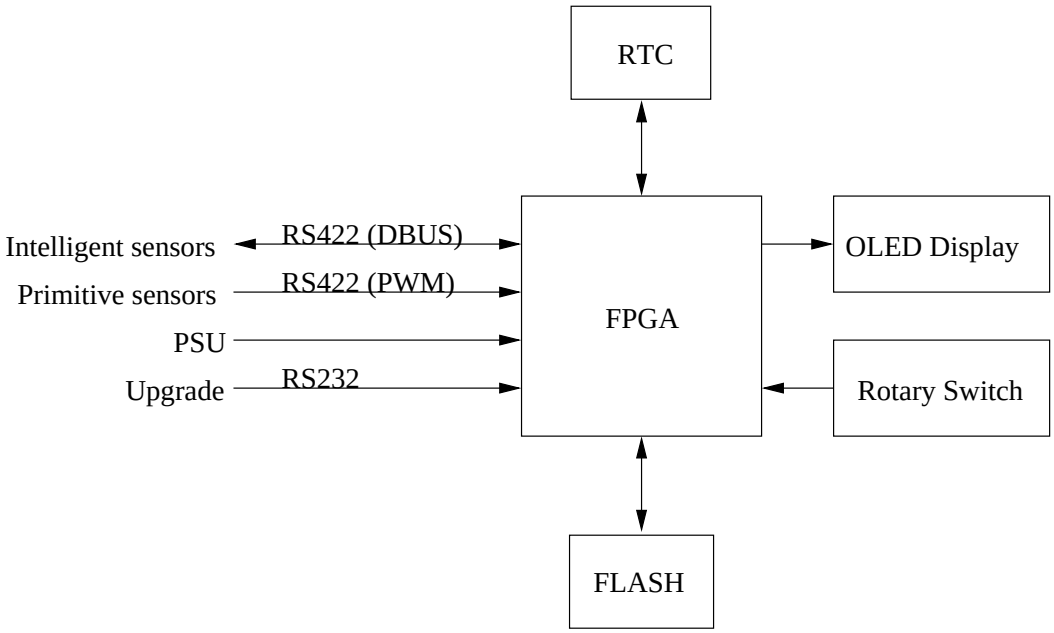

Figure 1: System overview

2.2 Subsystems

The system will be divided into the following subsystems: OLED unit, HSI/ADI/ESI/setup modes, and a rotary switch unit.

2.3 Design method

The system shall be built in modules with well defined interfaces to enable upgrade of a single subsystem independent of the others. The system shall

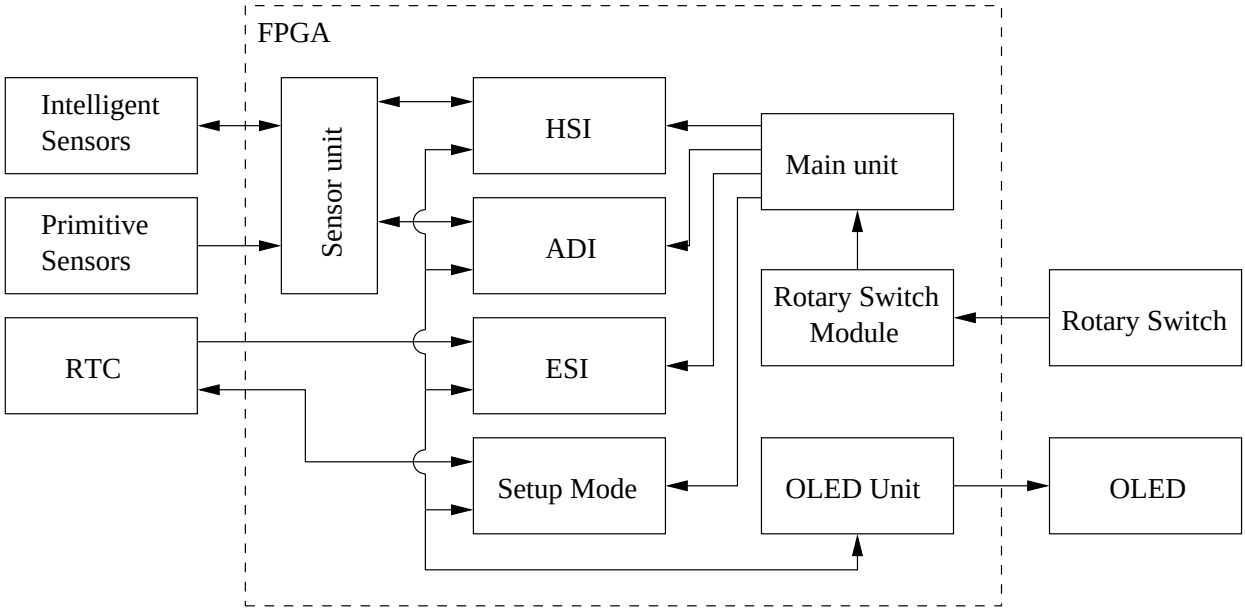

Figure 2: Inside FPGA

be developed with use of VHDL, if needed a softcore will be included. One single FPGA shall be used and one single FLASH memory.

3 Sensors

The back of the MEFIS casing has eight possible external connectors. One of these is intended for power supply, one for upgrades/maintenance of the software in the FPGA, and the other six for sensor units.

We will use two different kind of sensors; simple and intelligent. The simple sensors are pulse width modulated (PWM) and the intelligent work through a standard (DBUS) developed by the customer (DST Control AB). We will use the following sensor units:

- Magnetic sensor unit (DBUS)
- Outside air temperature (PWM)
- Pneumatic sensor unit (optional)
- Engine sensor unit (optional)
- Electric sensor unit (optional)

Notice that all sensor units include several actual sensors.

3.1 Magnetic sensor unit

The magnetic sensor unit is an "intelligent" unit, which means that we can both read and write from/to this unit. From this unit we get the aircraft magnetic heading, from which we can measure the:

- Yaw angle and angular velocity
- Roll angle and angular velocity
- Pitch angle and angular velocity
- The magnetic heading (compass)

3.2 Outside air temperature

This unit simply supply the outside air temperature as measured by it's sensor

- Outside air temperature

3.3 Pneumatic sensor unit (optional)

The pneumatic sensor unit gives us the static and dynamic pressure from the aircraft pitot tube, which we can use to measure approximate values of the actual aircraft speed and altitude.

- Altitude
- Airspeed

3.4 Engine sensor unit (optional)

The engine sensor unit measures various engine parameters such as:

- Oil pressure
- Fuel flow
- Manifold pressure
- Engine speed (RPM)
- Oil temperature

3.5 Electric sensor unit (optional)

The electric sensor unit measures the status of the airplane battery:

- Battery voltage
- Alternator current
- Load current

4 Flash

The FLASH memory contains the configuration of the FPGA. The configuration will be downloaded to the FPGA every time the MEFIS is started. If needed by the Setup module, some of the remaining memory can be used for configuration parameters for the MEFIS.

5 RTC (real-time clock)

The real-time clock is programmable from the setup mode and is displayed in the ESI mode. It has four external pins for communication. In the clock itself, there are address fields that can be read and written. These contain the year, month, day, hour, minute and second of the clock.

5.1 Interface

The RTC communicates through the SPI interface. The external pins are SDI (Serial Data Input), SDO (Serial Data Output), CE (Chip Enable) and SCLK (Serial Clock).

6 ADI and HSI units

These modules are the modes displaying sensor data on the OLED display. These are functions are very similar and therefore described in the same section. They use the same functions with only different data. This section will describe these functions and how the modules are designed in detail, also the interfaces with other modules will be defined.

The HSI and ADI modules will make use of the following functions, where the # is exchanged for HSI and ADI respectively:

- ACTIVATION_#

- INACTIVATION_#
- SEND_OLED
- INIT_OLED_#
- SEND_SENSOR_DATA_#

6.1 Description

The HSI and ADI modules is very similar and uses the same functions. The only differences is in when initiliazing the modules different characters are sent and that they collect different sensor data. The communication with other modules are similar. They collect data from the sensor unit, which is sampling with frequency 50 Hz, converts the data to characters and send it to the OLED unit.

6.2 ACTIVATION function

The ACTIVATION function synchronizes the *active* signal recieved from the Main Unit. It also produces a clock signal *clkActive* that is low when module is inactive and follows *clk* when module is active. See figure 3.

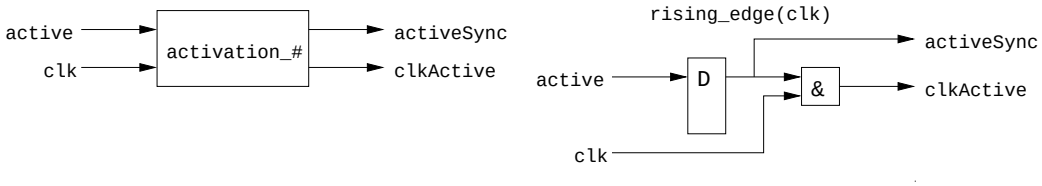

Figure 3: ACTIVATION_# overview

6.3 INACTIVATION function

The INACTIVATION function just set *init* low so that the INIT_OLED function starts the next time the module becomes active. See figure 4.

6.4 SEND_OLED function

The SEND_OLED function sends a singel character to the OLED when requested. When *sendRequest* is high, the *id* and *id* is send to OLED as described in figure 5. The characters are coded as ASCII.

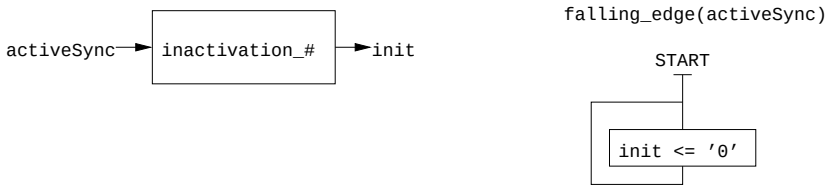

Figure 4: INACTIVATION_# overview

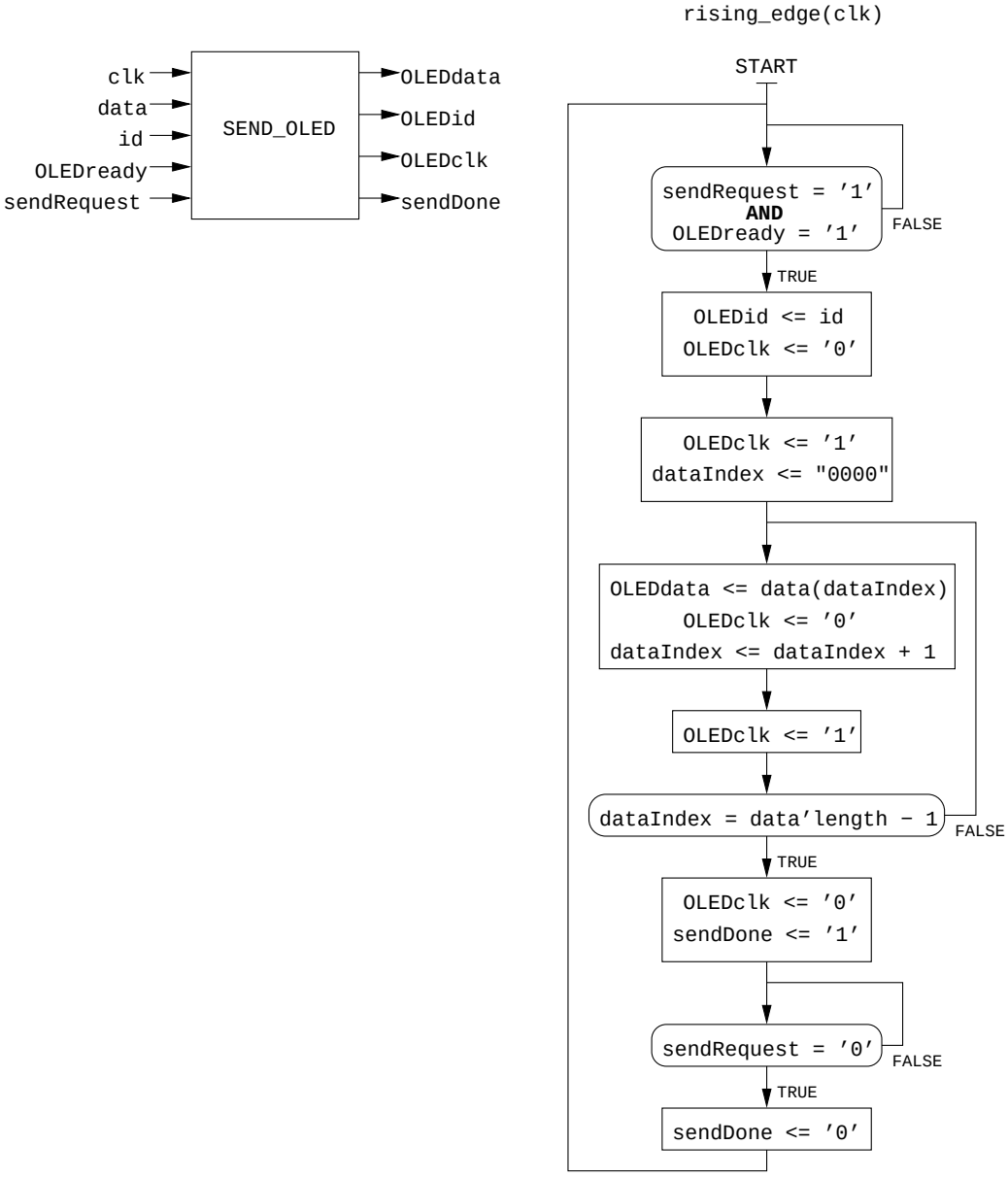

Figure 5: SEND_OLED overview

6.5 INIT_OLED function

The INIT_OLED function runs when the main unit switches on the module. It first clears the screen and then, one by one, sends out the characters. See figure 6.

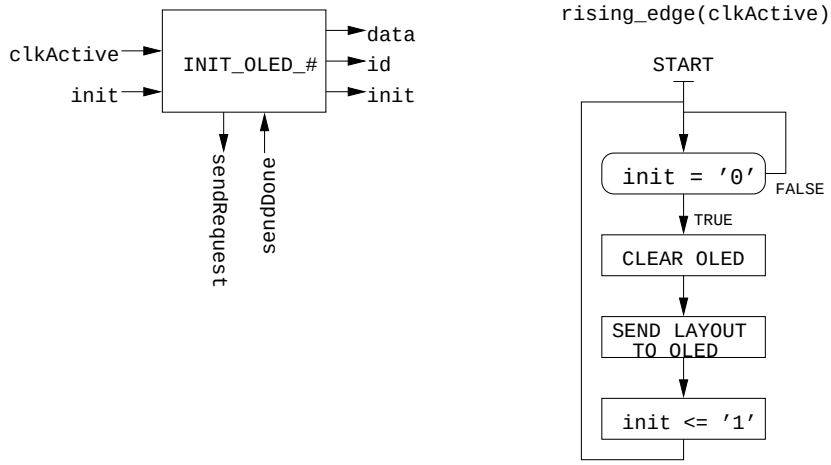

Figure 6: INIT_OLED_# overview

6.6 SEND_SENSOR_DATA_# function

The SEND_SENSOR_DATA function is first collecting the needed data when the signal *newData* gets high. For the ADI unit the *magneticHeading* and *outsideAirTemperature* is collected, and the HSI unit collects the *rollAngle, pitchAngle, slip, outsideAirTemperature* and the *verticalSpeed*. Then the data is splitted into single digits so it can be sent with the SEND_OLED function. See figure 7.

7 Sensor Unit

This block is the ADI and HSI unit way to communicate with the sensors.

7.1 Description

The sensor unit is going to sample the sensor data with frequency 50 Hz. It communicates with the intelligent sensors, in our case the ????, through DBUS which is developed by DST. Some education must be performed to

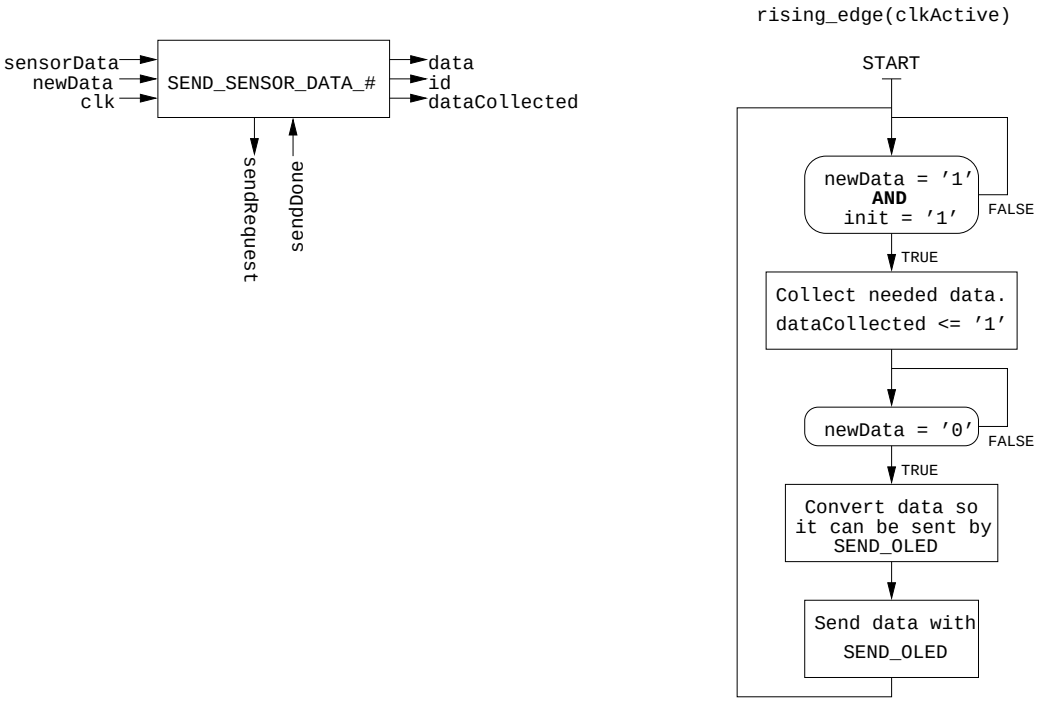

Figure 7: SEND_SENSOR_DATA_# overview

understand the usage of the VHDL blocks that DBUS consists of. The primitive sensors, in our case the outside air temperature, will continually send data via pulse modulator.

The unit will consist of the following functions

- SAMPLE_PERIOD
- GET_INTELLIGENT_DATA
- GET_PRIMITIVE_DATA

7.2 SAMPLE_PERIOD function

The SAMPLE_PERIOD function will send out a signal *getData* with frequency 50 Hz.

7.3 GET_INTELLIGENT_DATA function

The GET_INTELLIGENT_DATA function ...

7.4 GET_PRIMITIVE_DATA function

The GET_PRIMITIVE_DATA function will ...

8 ESI Unit

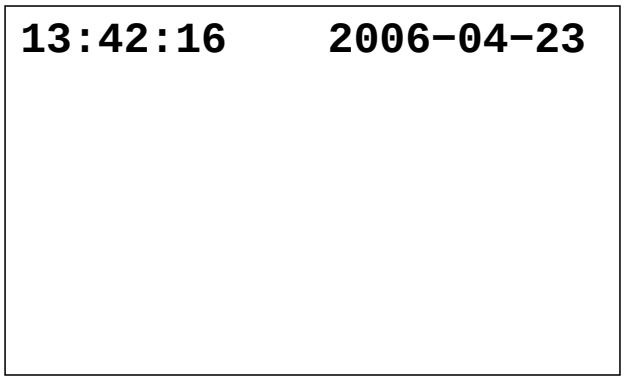

Figure 8: ESI unit

8.1 Description

This mode will display the date and time, similarly to the setup unit. This is a separate unit because there shall be future upgrades available for it. These upgrades include:

9 Setup Unit

9.1 Description

In this mode the pilot will be able to set time and date of the RTC, which will be displayed in ESI mode. The setup mode will work according to figure 9

9.2 10HOURS

After pushing the button one time in Setup mode, it will enter the sub-mode 10HOURS. This block will read the 10Hours address from the RTC and store it in the variable hour_h, see figure 10. This variable will only be 0, 1 or 2.

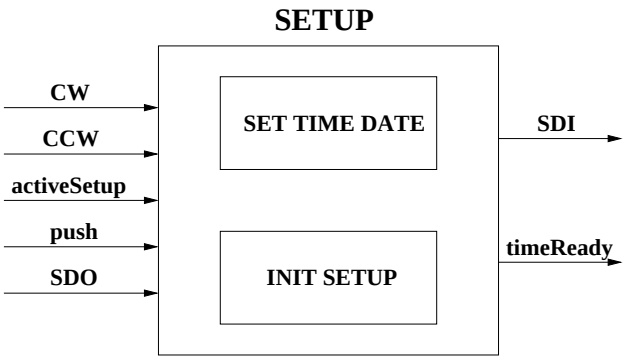

Figure 9: Setup unit

9.2.1 INC(10HOURS)

If the user rotates the switch clockwise, this block increases the variable hour_h by one.

9.2.2 DEC(10HOURS)

If the user rotates the switch counter-clockwise, this block decreases the variable hour_h by one.

9.2.3 SEND OLED(10HOURS)

This block will firstly make sure the handshake signal (Ready signal) from the OLED is good (=1). If this is the case it will update the 10hours part of the time display on the display. It does this by sending a 3 bit ID (which is "Character", this is a constant explained in the OLED chapter), and after this setting the start coordinate (both horizontal and vertical coordinates, these are constants). Following this is the hour_h variable.

9.3 HOURS

After pushing the button the second time in Setup mode, it will enter the sub-mode HOURS. This sub-mode will read the Hours address from the RTC and store it in the variable hour_l. This variable will be 0 to 9.

9.3.1 INC(HOURS)

If the user rotates the switch clockwise, this block increases the variable hour_l by one.

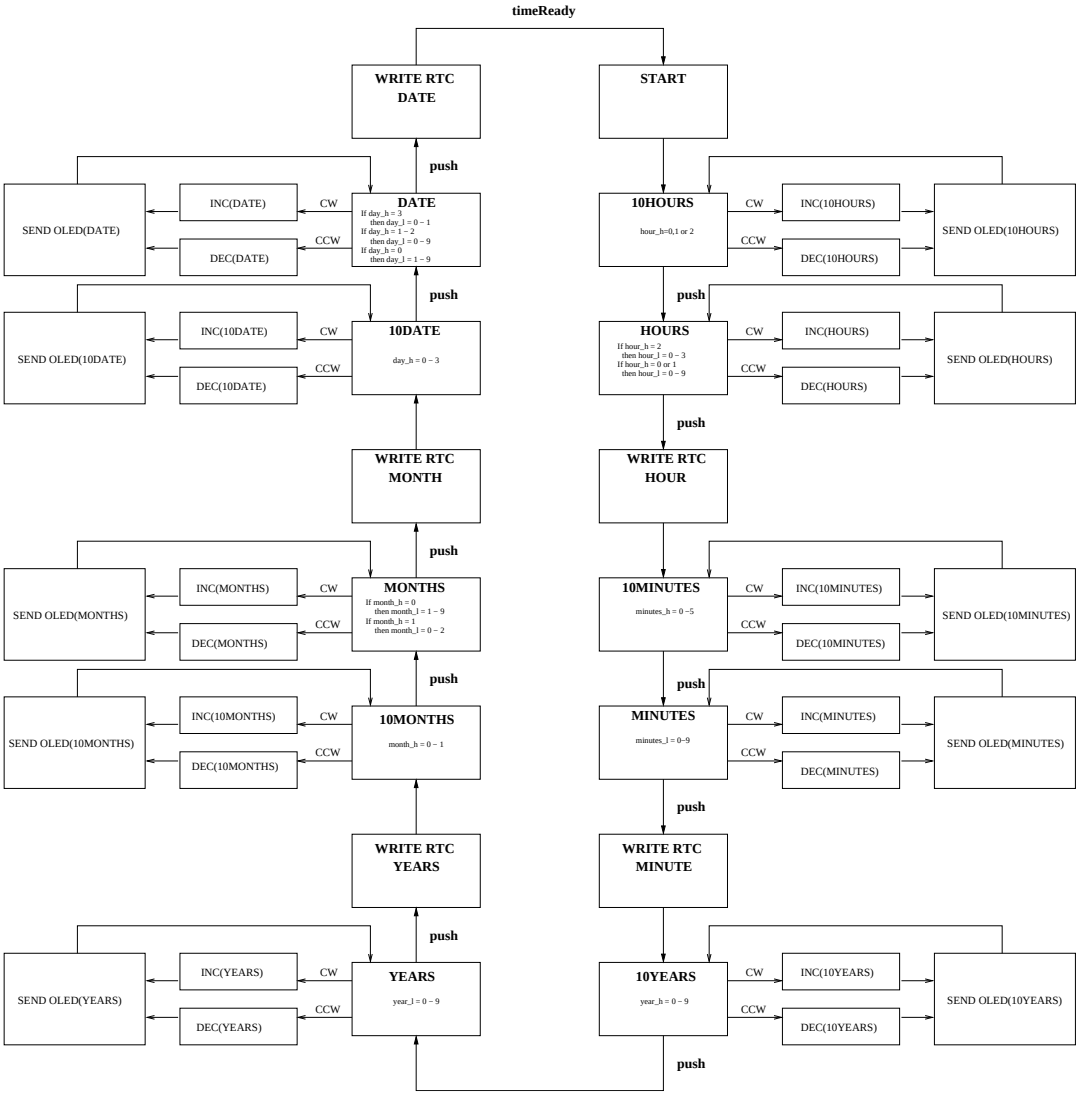

Figure 10: Function of SET TIME DATE

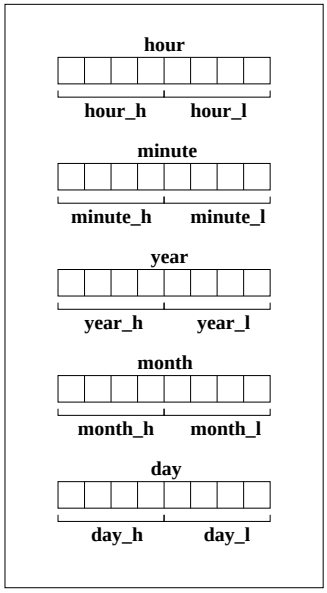

Figure 11: Variables of SET TIME DATE

9.3.2 DEC(HOURS)

If the user rotates the switch counter-clockwise, this block decreases the variable hour_l by one.

9.3.3 SEND OLED(HOURS)

This block will firstly make sure the handshake signal (Ready signal) from the OLED is good (=1). If this is the case it will update the hours part of the time display on the display. It does this by sending a 3 bit ID (which is "Character", this is a constant explained in the OLED chapter), and after this setting the start coordinate (both horizontal and vertical coordinates, these are constants.). Following this is the hour_l variable.

9.4 WRITE RTC(HOUR)

When pushing the button the third time in Setup mode, the hours will be written into the appropriate memory address of the RTC. After this the setup mode will enter sub-mode MINUTES. This, and all the other sub-modes in the flow-diagram, work similarly to the HOURS sub-mode. described above.

The following settings will be **optional**:

- Set the barometric pressure (provided by air traffic control)

- Turn off some functions in HSI and ADI mode
- A possibility to select between two light intensity modes: normal and dimmed, and to adjust the intensity in these modes

9.5 Interface

The RTC will communicate with setup mode using SPI interface.

10 Rotary switch unit

10.1 Description

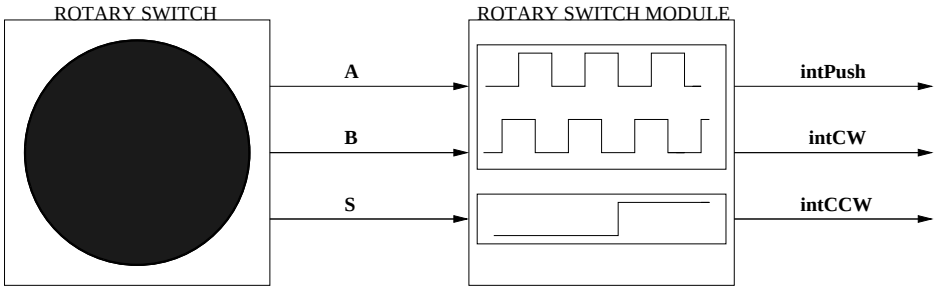

Figure 12: The rotary switch unit

This unit utilizes a button integrated with a rotary switch and decoding software. The button is model EM11B from the manufacturer ALPS.
The following functions shall be detectable by the rotary switch unit and will be outputs to the Main unit.

- Single tap
- Rotary switching to the left and right
- Double tap (optional)
- Holding down the button (optional)

11 Main unit

11.1 Description

This unit will handle which mode that is active. Input signal is the interrupt from rotary switch unit. This will be set by a flag (flagActive) that will be

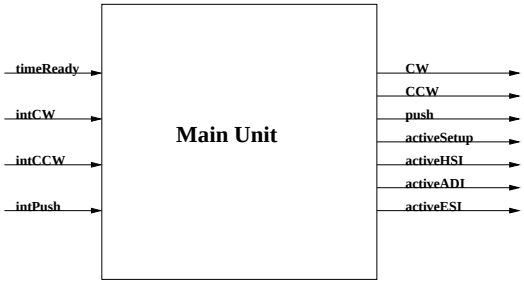

Figure 13: Main unit

any of the numbers 1 to 4 as specified below.

11.2 Setup Mode

If the setup mode is active, the main unit must also check that the signal *timeReady* is high. If *timeReady* is high the main unit will work as usual. However, if *timeReady* is low, this means that the user is setting date and time. In this case the main unit will send the signals *intCW* and *intCCW* from the rotary switch module straight to the setup. The signal *intPush* will always be forwarded to the outgoing signal *push* in setup mode.

Active mode	**intCW**	**intCCW**	**intPush**
Setup (1)	if timeReady = 1 flagActive + 1 else CW = intCW	if timeReady = 1 flagActive = 4 else CCW = intCCW	push = intPush
HSI (2)	flagActive + 1	flagActive -1	ignore
ADI (3)	flagActive + 1	flagActive -1	ignore
ESI (4)	flagActive = 1	flagActive -1	ignore

12 OLED

This module is supposed to act as a graphic generator and present the data sent by HSI, ADI, ESI and the Setup modules on the display. The display hardware has none of the features as character, line or circle generators. All these will, if time allows, be implemented in the FPGA in the module OLED. For the ease of upgrade and module block thinking the data received from the other blocks will be sent in a particular order. More about the packet organisation can be found under the subsection *Data receiver*. This section will describe in detail how the module will be designed and specify interfaces with its surrounding.

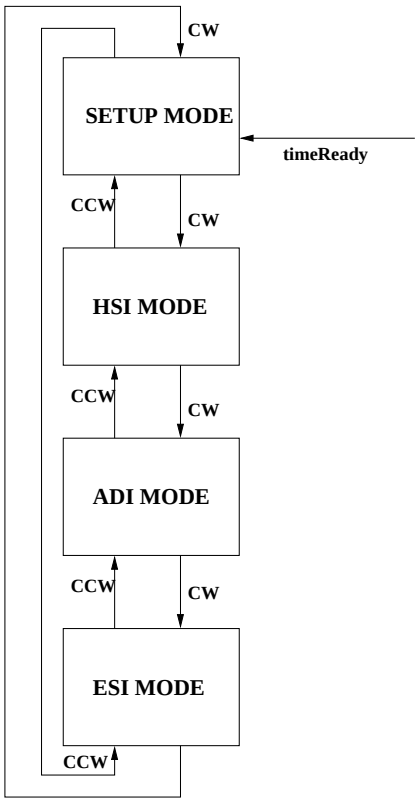

Figure 14: The order in which the modes switch

The OLED module is divided into the following seven blocks:

- Data receiver
- Clear
- Character generator
- Line generator (optional)
- Circle generator (optional)
- Initiator
- Data transmitter

12.1 Rough description of the module

The data that are to be displayed will be recieved from the other units by a *Data reciever* block. This data will be sent as defined in the subsection *Data receiver*. The type of data, refered to as the data ID, will be defined by four bits in a special register and will decide which block to process the data.

There are five diffrent blocks that handle different kind of datatypes (different data ID). These blocks are; *Clear* that will clear the display when switching mode, *Line generator* that will draw lines, *Circle generator* that will draw circles, *Character generator* that will print different characters and *Initiator*, that will initiate the display when reset. When any of these blocks has finished its task it will signal ready to the *Data receiver* that in turn will signal ready to the transmitting unit and enable new data to be transmitted and processed.

The blocks will send their information to the display circuit of type SSD0323 trough a *Data transmitter* block. This block will perform all nescessary steps to enable communication with the OLED.

12.2 Data receiver

This block will receive data from the ESI, HSI, ADI and the Setup modules. The data is received in a particular order as defined below.

If the *Data receiver* (DR) is ready to receive data, the ready pin is pulled high. If the module which is connected to the Data receiver want to send data, the module first set the three ID-pins corresponding to the data to be transmitted. Thereafter one clock pulse is sent (from the transmitter) and the ID is stored in a 4-bit register (in the DR), and the ready pin is pulled

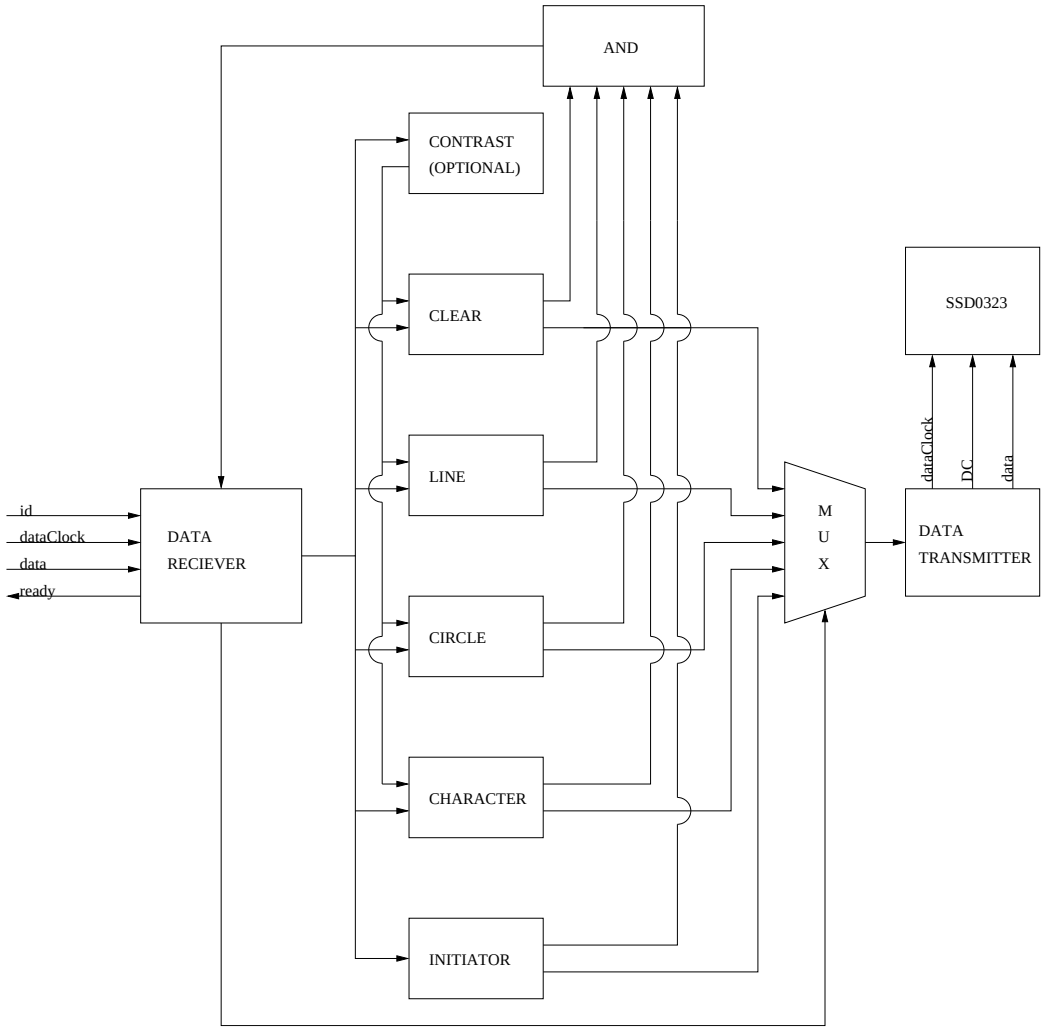

Figure 15: Overview of the OLED unit

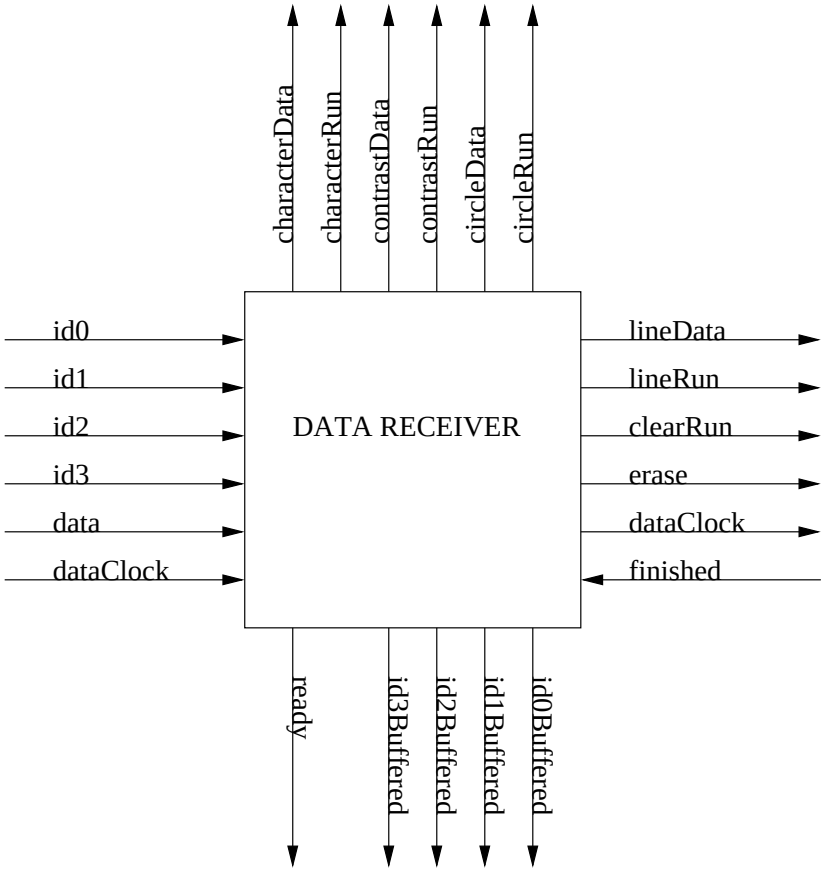

Figure 16: Outline of the Data receiver

low (by DR). The DR's multiplexer is switched to the corresponding block by ID and is ready for block specific data. On the next clock pulse (from the transmitter) the first data bit is received on the data pin. When all the specified bits (by ID) are stored the DR sends a start pulse to the corresponding block; character, line, circle or clear. The Data receiver then waits for the tasks to complete, by waiting for a ready signal from the specified block.
The following ID's will be used:

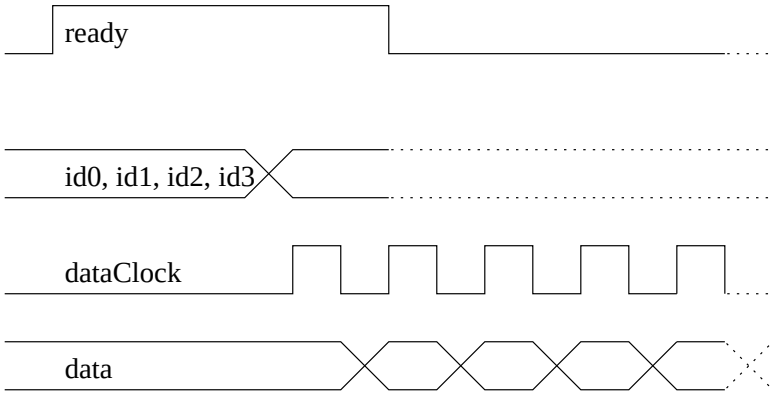

Figure 17: How to access the Data receiver

ID	ID bits
Init	0000
Clear	0001
Character	0010
Line	0011
Anti line	0100
Circle	0101
Anti circle	0110

The start coordinate is the same for all the blocks that must have a start coordinate. It is specified by 6 + 7 bits, which is the row (64 rows) and column (128 columns) respectively.

12.3 Clear

When this block is adressed by ID and started, the whole display is cleared. The block does not require any data but ID and is started just after ID is stored in the Data receiver's 4-bit ID register. When finished the block sends a signal, clear_ready, to the Data receiver block to state that the task has been processed. The Data receiver is then ready to process the next command.

In the following text the data sent is assumed to be sent to the Data transmitter before the display. When the Clear block receives a start pulse the command "Set Column Address" is sent to goto the display line zero. Then the command "Set Row Address" is sent to goto the display row zero. Then 128x64 pixel data containing zeros is sent to clear the whole display. Clear_ready is then sent to let the next task proceed.

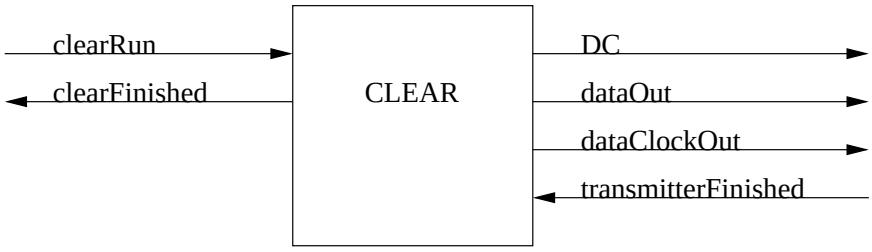

Figure 18: Outline of the Clear block

12.4 Contrast

The contrast block holds the contrast that will be used for all the pixels on the display. When reset the register that holds the contrast value is set to a default value. If the value is changed, in *Setup mode*, the *Setup module* sends the new contrast value through the *Data receiver* to the contrast block. The contrast is changeable (16 different values) and is stored in a 4-bit register.

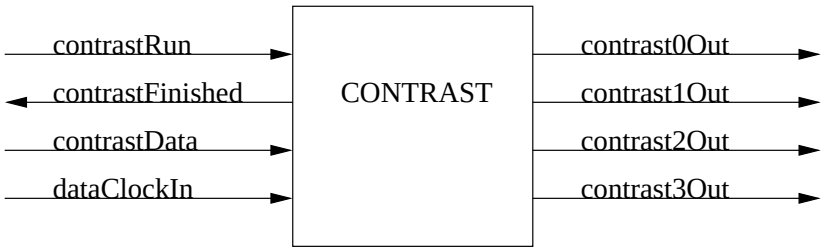

Figure 19: Outline of the Contrast block

12.5 Character generator

If the data received is of character type this block will get the corresponding character from a character memory and send it to the *Data transmission* block. The characters are predefined and stored in the FPGA. All characters are of the same height and width (monospace).

This block contains a register that is adressed by the Data receiver when a character is to be displayed. When the register is filled with data, start coordinates and the specific character code, and receives a start pulse from the *Data receiver*, the character is sent via the emphData transmitter to the display. When finished the block sends a signal, character_ready, to the *Data receiver* to state that the task has been processed. The *Data receiver* is then ready to process the next command.

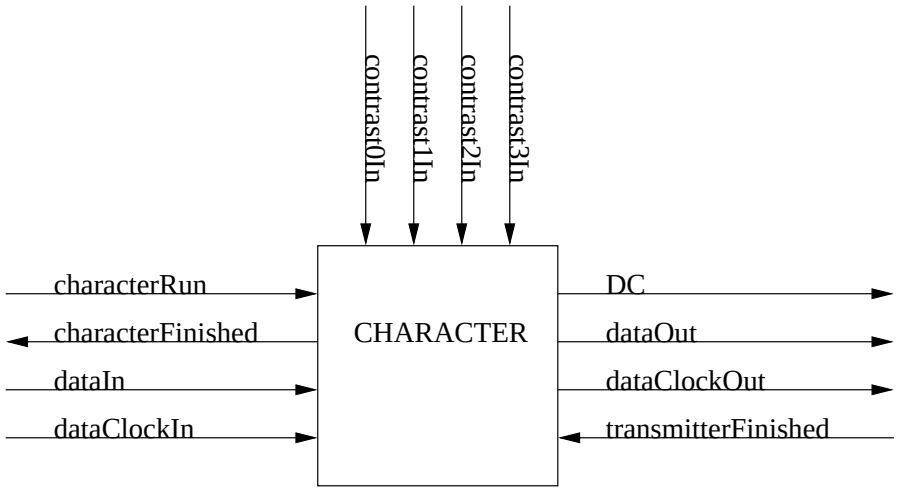

Figure 20: Outline of the Character block

The specific character code is sent after the mandatory start coordinate and tells the character generator which character to get from the memory. The character code is specified by 8 bits. The character is then displayed (via the *Data transmitter*) on the OLED with respect to the upper left corner as reference. The upper left corner of the character will be put at the particular point stated by the start coordinate.

In the following text the data sent is assumed to be sent to the *Data transmitter* before the display. When the Character generator receives a start pulse the command "Set Column Address" is sent to goto the start line, and to set the character width. Then the command "Set Row Address" is sent to goto the start row, and to set the character height. The Character generator then gets the right character from memory and shovels it out to the display. Character_ready is then sent to *Data receiver* to let the next task proceed.

12.6 Line generator (optional)

If the data received is of line type this block will compute the pixels that are to be lit and send it to the *Data transmitter* block. This block contains a register that is adressed by the *Data receiver* when a line is to be displayed. When the register is filled with data, start and stop coordinates, and receives a start pulse from the *Data receiver*, the line pixels are computed and is sent via the Data transmitter to the display. When finished the block sends a signal, line_ready, to the *Data receiver block* to state that the task has been

processed. The *Data receiver* is then ready to process the next command.

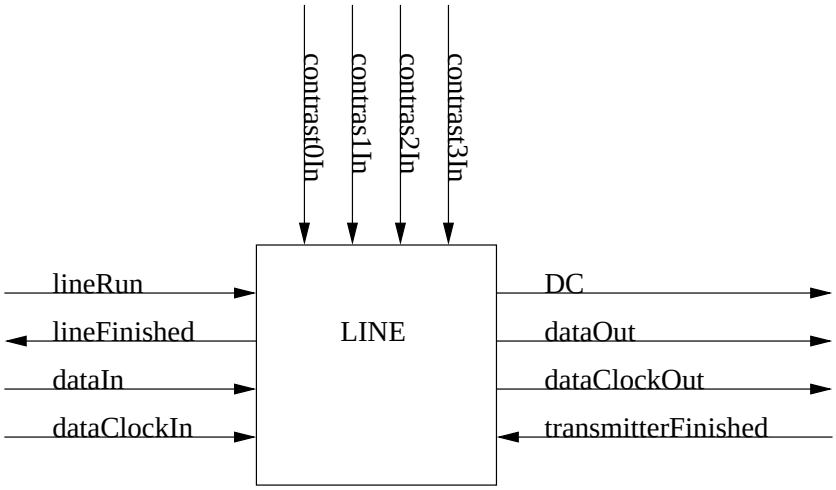

Figure 21: Outline of the Line block

The *Line generator* receives, apart from the start coordinate that is mandatory, a stop coordinate that tells the block where to end the line. The stop coordinate is specified by 6 + 7 bits, which is the row (64 rows) and column (128 columns) respectivly. The start coordinate is sent first and then the stop coordinate.

12.7 Circle generator (optional)

If the received data is of circle type this block will compute the pixels that are to be lit and send it to the *Data transmitter* block. The *circle generator block* contains a register that is adressed by the *Data receiver* when a circle is to be displayed. When the register is filled with data containing start coordinates and the circle radius, and receives a start pulse from the *Data receiver*, the circle pixels are calculated and sent via the Data transmitter to the display. When finished the block sends a signal, circle_ready, to the *Data receiver block* to state that the task has been processed. The *Data receiver* is then ready to process the next command.

The *Circle generator* receives, apart from the start coordinate that is mandatory, a radius for the circle. The radius is specified by 7 bits as the circle can have a radius of 128 columns. The start coordinate is sent first and the the radius.

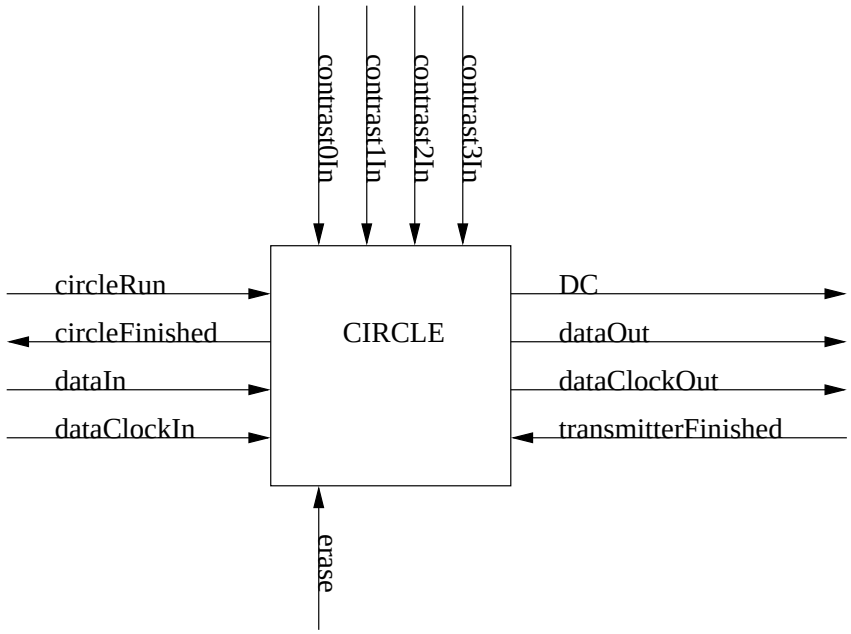

Figure 22: Outline of the Circle block

12.8 Initiator

This block is active when the circuit is reset and initiates the display for correct operation. When the *Data receiver* is reset the ID register contains 0000 which is the adress code for the Initiator. A start pulse is received and the Initiator starts to send commands via the Data transmitter to the display. The commands set the display hardware to the right state. When finished the block sends a signal, initiator_ready, to the Data receiver block to state that the task has been processed. The Data receiver is then ready to process the next command.

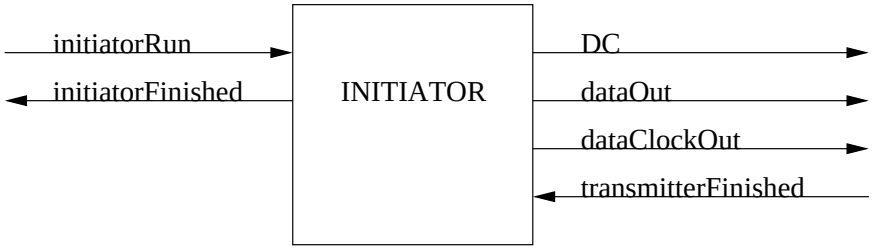

Figure 23: Outline of the Initiator block

The Initiator will set the following pins on the SSD0323 circuit (not necessarily in this order):

- CL: External clock input, maximal possible frequency is 4 MHz.
- M/S#: "1"
- CLS: "0" - To disable internal clock.
- BS0, BS1, BS2: "000" - To set serial interface mode.
- RES#: "0" - Initiates the display.

The following commands will be sent to the SSD0323 circuit:

- Set Contrast Control Register
- Set Full Current Range
- Set Re-map
- Set Display Offset
- Set Multiplex Ratio
- Set Phase Length
- Set Row Period
- Set V_{COMH} Voltage
- Set Precharge Voltage
- Set Segment Low Voltage
- Set Segment Low Voltage

12.9 Data transmitter

This block receives data from the blocks *character generator, line generator, circle generator* and *initiator* and forwards it to the display. The procedure of sending will be as follows.

- Set D/C# to the appropiate state (0=command 1=data).
- Output "Data clock out" on the CL pin on the SSD0323.

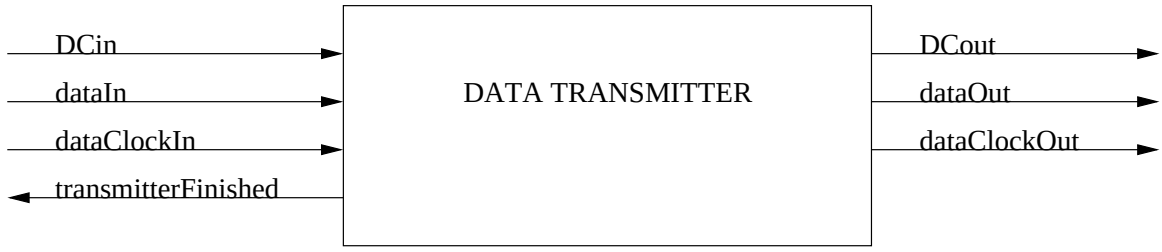

Figure 24: Outline of the Data transmotter block

- Set CS# to 0 to enable communication.
- Start shifting data from the register in the data transmission block with the minimum period possible (250 ns) into the SSD0323.
- When all data or commands are sent, send transmitter_ready to current sending block.

The data or commands that are to be sent will be transferred serial into a shift register in the data transmitter block. Also the D/C bit is needed. The data clock out will trigger the transmission to start.

References

[1] Tomas Svensson & Christian Krysander, *Projektmodellen Lips*, kompendium, Linköpings Tekniska Högskola, Version 1.2.

[2] George Grätzer, *Math into LaTeX*, Birkhäuser, 1996.

[3] Jan-Erik Strömberg, *Requirement specification – Micro EFIS*, DST Control, MEFIS/Doc/Spec/R0543S01.fm, version 1.0.0.