

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 "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

E·XFI

Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega6490-16ai

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong



priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). Refer to "Interrupts" on page 49 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see "Boot Loader Support – Read-While-Write Self-Programming" on page 278.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

	n r16, SREG ; store SREG value		
	ii ; disable interrupts during timed sequence		
	bi EECR, EEMWE ; start EEPROM write		
	bi EECR, EEWE		
	ut SREG, r16 ; restore SREG value (I-bit)		
СС	de Example		
	har cSREG;		
	cSREG = SREG; /* store SREG value */		
	* disable interrupts during timed sequence */		
	_disable_interrupt();		
	ECR = (1< <eemwe); *="" <="" eeprom="" start="" th="" write=""><th></th></eemwe);>		
	$EECR \mid = (1 << EEWE);$		
	REG = cSREG; /* restore SREG value (I-bit) */		

8.6 External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 8-3. To run the device on an external clock, the CKSEL Fuses must be programmed to "0000".





When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 8-10.

Table 8-9.	Crystal Oscillator Clock Frequency
------------	------------------------------------

CKSEL30	Frequency Range
0000	0 - 16MHz

 Table 8-10.
 Start-up Times for the External Clock Selection

SUT10	Start-up Time from Power- down and Power-save	Additional Delay from Reset (V _{CC} = 5.0V)	Recommended Usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4.1ms	Fast rising power
10	6 CK	14CK + 65ms	Slowly rising power
11		Reserved	

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to "System Clock Prescaler" on page 32 for details.

8.7 Clock Output Buffer

When the CKOUT Fuse is programmed, the system Clock will be output on CLKO. This mode is suitable when the chip clock is used to drive other circuits on the system. The clock will be output also during reset and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including internal RC Oscillator, can be selected when CLKO





12. External Interrupts

The External Interrupts are triggered by the INT0 pin or any of the PCINT30..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT30..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. The pin change interrupt PCI1 will trigger if any enabled PCINT15..8 pin toggles. Pin change interrupts PCI0 will trigger if any enabled PCINT7..0 pin toggles. The PCMSK3, PCMSK2, PCMSK1, and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT30..0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A – EICRA. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 requires the presence of an I/O clock, described in "Clock Systems and their Distribution" on page 26. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses as described in "System Clock and Clock Options" on page 26.

12.1 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 12-1.





⁵⁴ ATmega329/3290/649/6490



Table 13-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 13-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

 Table 13-2.
 Generic Description of Overriding Signals for Alternate Functions

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

Some pins are connected to different LCS segments on ATmega3290/6490 and ATmega3290/6490. See pinout on "Pinout ATmega3290/6490" on page 2 and "Pinout ATmega329/649" on page 3 for details.



• SS/PCINT8 – Port B, Bit 0

 \overline{SS} : Slave Port Select input. When the SPI is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB0. As a Slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a Master, the data direction of this pin is controlled by DDB0. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB0 bit

PCINT8, Pin Change Interrupt Source 8: The PB0 pin can serve as an external interrupt source.

Table 13-7 and Table 13-8 relate the alternate functions of Port B to the overriding signals shown in Figure 13-5 on page 65. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

	0 0			
Signal Name	PB7/OC2A/ PCINT15	PB6/OC1B/ PCINT14	PB5/OC1A/ PCINT13	PB4/OC0A/ PCINT12
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2A ENABLE	OC1B ENABLE	OC1A ENABLE	OC0A ENABLE
PVOV	OC2A	OC1B	OC1A	OC0A
PTOE	_	_	_	-
DIEOE	PCINT15 • PCIE1	PCINT14 • PCIE1	PCINT13 • PCIE1	PCINT12 • PCIE1
DIEOV	1	1	1	1
DI	PCINT15 INPUT	PCINT14 INPUT	PCINT13 INPUT	PCINT12 INPUT
AIO	-	_	-	_

Table 13-7. Overriding Signals for Alternate Functions in PB7:PB4

14. 8-bit Timer/Counter0 with PWM

14.1 Features

Timer/Counter0 is a general purpose, single compare unit, 8-bit Timer/Counter module. The main features are:

- Single Compare Unit Counter
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- External Event Counter
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV0 and OCF0A)

14.2 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 14-1. For the actual placement of I/O pins, refer to "Pinout ATmega3290/6490" on page 2 and "Pinout ATmega329/649" on page 3. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the "Register Description" on page 103.





14.2.1 Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0A) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter





Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	ТОР	Update of OCR0A at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	СТС	OCR0A	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

 Table 14-2.
 Waveform Generation Mode Bit Description⁽¹⁾

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

• Bit 5:4 – COM0A1:0: Compare Match Output Mode

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM01:0 bit setting. Table 14-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM).

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match
1	0	Clear OC0A on compare match
1	1	Set OC0A on compare match

 Table 14-3.
 Compare Output Mode, non-PWM Mode

Table 14-4 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

 Table 14-4.
 Compare Output Mode, Fast PWM Mode⁽¹⁾

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Reserved
1	0	Clear OC0A on compare match, set OC0A at BOTTOM, (non-inverting mode)
1	1	Set OC0A on compare match, clear OC0A at BOTTOM, (inverting mode)

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 98 for more details.

Table 14-5 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to phase correct PWM mode.



16.11.9 TIFR1 – Timer/Counter1 Interrupt Flag Register



• Bit 5 – ICF1: Timer/Counter1, Input Capture Flag

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 Flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

• Bit 2 – OCF1B: Timer/Counter1, Output Compare B Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a Forced Output Compare (FOC1B) strobe will not set the OCF1B Flag.

OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

Bit 1 – OCF1A: Timer/Counter1, Output Compare A Match Flag

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

Bit 0 – TOV1: Timer/Counter1, Overflow Flag

The setting of this flag is dependent of the WGM13:0 bits setting. In Normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to Table 16-5 on page 134 for the TOV1 Flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.



When OC2A is connected to the pin, the function of the COM2A1:0 bits depends on the WGM21:0 bit setting. Table 17-3 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to a normal or CTC mode (non-PWM).

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Toggle OC2A on compare match.
1	0	Clear OC2A on compare match.
1	1	Set OC2A on compare match.

 Table 17-3.
 Compare Output Mode, non-PWM Mode

Table 17-4 shows the COM2A1:0 bit functionality when the WGM21:0 bits are set to fast PWM mode.

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Reserved
1	0	Clear OC2A on compare match, set OC2A at BOTTOM, (non-inverting mode).
1	1	Set OC2A on compare match, clear OC2A at BOTTOM, (inverting mode)

 Table 17-4.
 Compare Output Mode, Fast PWM Mode⁽¹⁾

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at BOTTOM. See "Fast PWM Mode" on page 146 for more details.

Table 17-5 shows the COM21:0 bit functionality when the WGM21:0 bits are set to phase correct PWM mode.

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Reserved
1	0	Clear OC2A on compare match when up-counting. Set OC2A on compare match when counting down.
1	1	Set OC2A on compare match when up-counting. Clear OC2A on compare match when counting down.

 Table 17-5.
 Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

Note: 1. A special case occurs when OCR2A equals TOP and COM2A1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 148 for more details.



18. SPI – Serial Peripheral Interface

18.1 Features

The ATmega329/3290/649/6490 SPI includes the following features:

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

18.2 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega329/3290/649/6490 and peripheral devices or between several AVR devices. A simplified block diagram of the Serial Peripheral Interface is shown in Figure 18-1.

The PRSPI bit in "Power Reduction Register" on page 37 must be written to zero to enable the SPI module.



Figure 18-1. SPI Block Diagram⁽¹⁾

Note: 1. Refer to Figure 1-1 on page 2, and Table 13-6 on page 68 for SPI pin placement.

158 ATmega329/3290/649/6490

22.7 Register Description

22.7.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7:6 - REFS1:0: Reference Selection Bits

These bits select the voltage reference for the ADC, as shown in Table 22-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 22-3.	Voltage Reference Selections for AE	C
-------------	-------------------------------------	---

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

• Bit 5 – ADLAR: ADC Left Adjust Result

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "ADCL and ADCH – The ADC Data Register" on page 226.

Bits 4:0 – MUX4:0: Analog Channel Selection Bits

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 22-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).





Bit Number	Signal Name	Module
157	PE0.Data	Port E
156	PE0.Control	
155	PE0.Pull-up_Enable	
154	PE1.Data	
153	PE1.Control	
152	PE1.Pull-up_Enable	
151	PE2.Data	
150	PE2.Control	
149	PE2.Pull-up_Enable	
148	PE3.Data	
147	PE3.Control	
146	PE3.Pull-up_Enable	
145	PE4.Data	
144	PE4.Control	
143	PE4.Pull-up_Enable	
142	PE5.Data	
141	PE5.Control	
140	PE5.Pull-up_Enable	
139	PE6.Data	
138	PE6.Control	
137	PE6.Pull-up_Enable	
136	PE7.Data	
135	PE7.Control	
134	PE7.Pull-up_Enable	

Table 25-7. ATmega329/649 Boundary-scan Order, 64-pin (Continued)

used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

26.7.11 Programming Time for Flash when Using SPM

The calibrated RC Oscillator is used to time Flash accesses. Table 26-5 shows the typical programming time for Flash accesses from the CPU.

Table 26-5. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (Page Erase, Page Write, and write Lock bits by SPM)	3.7ms	4.5ms



- 1. Set XA1, XA0 to "00". This enables address loading.
- 2. Set BS1 to "1". This selects high address.
- 3. Set DATA = Address high byte (0x00 0xFF).
- 4. Give XTAL1 a positive pulse. This loads the address high byte.
- H. Program Page
- 1. Give WR a negative pulse. This starts programming of the entire page of data. RDY/BSY goes low.
- 2. Wait until RDY/BSY goes high (See Figure 27-3 for signal waveforms).

I. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

J. End Page Programming

- 1. 1. Set XA1, XA0 to "10". This enables command loading.
- 2. Set DATA to "0000 0000". This is the command for No Operation.
- 3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.





Note: 1. PCPAGE and PCWORD are listed in Table 27-10 on page 298.



28.7 ADC Characteristics

Table 28-7.	ADC Characteristics
-------------	---------------------

Symbol	Parameter	Condition	Min	Тур	Мах	Units
		Single Ended Conversion		10		Bits
	Resolution	Differential Conversion		8		Bits
		Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 200kHz		2	2.5	LSB
		Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 1MHz		4.5		LSB
	Absolute accuracy (Including INL, DNL, quantization error, gain and offset error)	Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 200kHz Noise Reduction Mode		2		LSB
		Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 1MHz Noise Reduction Mode		4.5		LSB
	Integral Non-Linearity (INL)	Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 200kHz		0.5		LSB
	Differential Non-Linearity (DNL)	Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 200kHz		0.25		LSB
	Gain Error	Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 200kHz		2		LSB
	Offset Error	Single Ended Conversion $V_{REF} = 4V, V_{CC} = 4V,$ ADC clock = 200kHz		2		LSB
	Conversion Time	Free Running Conversion	13		260	μs
	Clock Frequency	Single Ended Conversion	50		1000	kHz
AVCC	Analog Supply Voltage		V _{CC} - 0.3		V _{CC} + 0.3	V
N		Single Ended Conversion	1.0		AVCC	V
VREF	Reference voltage	Differential Conversion	1.0		AVCC - 0.5	V
	Din Input Voltage	Single Ended Channels	GND		V _{REF}	V
V	Fin input voitage	Differential Channels	GND		AVCC	V
V IN	Input Panga	Single Ended Channels	GND		V _{REF}	V
	input hange	Differential Channels ⁽¹⁾	-0.85V _{REF}		V _{REF}	V
	Innut Donelusidal	Single Ended Channels		38.5		kHz
		Differential Channels		4		kHz





Figure 29-2. Active Supply Current vs. Frequency (1 - 16MHz))



Figure 29-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8MHz)





29.0.9 Pin Thresholds and hysteresis





Figure 29-34. I/O Pin Input Threshold Voltage vs. V_{CC} (V_{IL}, I/O Pin Read as "0")





29.0.11 Internal Oscillator Speed



Figure 29-45. Watchdog Oscillator Frequency vs. V_{CC}









29.0.13 Current Consumption in Reset and Reset Pulsewidth

Figure 29-56. Reset Supply Current vs. V_{CC} (0.1 - 1.0MHz, Excluding Current Through The Reset Pull-up)





31. Instruction Set Summary

PARTNER Description Per-Mail ZCAN H I ADD PR 4P Action Register Per-Mail And C ZCAN H I ADQ PRLK Add Immedias UV void C Bahda - Bahda L N ZCAN H I ADQ PRLK Additention Register Bel - Rei FR ZCAN H I BLB PRLK Additention Register Bel - Rei FR ZCAN H I BLD RLK Submain Terminic	Mnemonics	Operands	Description	Operation	Flags	#Clocks	
AOC R.A. P. Add to Register Pater Mat Proc ZON/H 1 AOC Pater Mat Add to Register Pater Mat Proc ZON/H 1 ADW Pater Mat Proc ZON/H 1 1 ADW Pater Mat Proc ZON/H 1 1 ADW Pater Mat Proc ZON/H 1 1 SUB Pater Mat Add to Constant from Register Pater Mat K ZON/H 1 BAD Pater Mat K Subtrast Mat Carry Constant from Page Pater Mat K ZON/H 1 BAD Pater Mat K Logical ADM Register and Constant from Page Pater Mat K ZON/H 1 ADM Pater K Logical ADM Register and Constant Pater Mat K ZNV 1 ADM Pater K Logical OR Register and Constant Pater Mat K ZNV 1 ADM Pater K Logical OR Register and Constant Pater Mat K ZNV 1 ADM Pater K Logical OR Register and Constant Pater Mat K ZNV 1	ARITHMETIC AND L	OGIC INSTRUCTION	s				
ADG Rd, K And encodiants tword Ref. Her. 40. Her. 60. ZO.NY M 1 SUB Rd, K Substat Constitution Register Ref. Ref. K ZO.NY M 1 SUB Rd, K Substat Constitution Register Ref. Ref. K ZO.NY M 1 SUB Rd, K Substat Constitution Register Ref. Ref. K-C ZO.NY M 1 SUB Rd, K Substat Constitution Register Ref. Ref. K-C ZO.NY M 1 SUB Rd, K Substat Constitution Register Ref. Ref. K-C ZO.NY M 1 SUM Rd, K Substat Constitution Register Ref. Ref. K-C ZO.NY M 1 SUM Rd, K Logizt ADD Register and Constitution Ref. Ref. K-C ZO.NY M 1 SUM Rd, K Logizt ADD Register and Constitution Ref. Ref. Ref. K-C ZO.NY M 1 SUM Rd, K Logizt ADD Register and Constitution Ref. Nef. Nef. AC ZO.NY M 1 SUM Rd, K Substat Constitution Register Ref. Nef. Nef. AC ZO.NY M	ADD	Rd. Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z.C.N.V.H	1	
ACW Rat. Act Immediate to Work Ram Rel - Ro-Brank ZON VII 1 SUB Rd, Kr Substant too Register Rd - FM - Kr ZON VII 1 SUB Rd, K Substant thoo Register Rd - FM - Kr ZON VII 1 SUB Rd, K Substant thoo Register Rd - FM - Kr ZON VII 1 SRC Rd, K Substant thoo Register Rd - FM - Kr ZON VII 1 SRC Rd, K Logical ADD Register and Constant Rd - FM - Kr ZON VI 1 AND Rd, K Logical ADD Register and Constant Rd - FM - KR ZON VI 1 CR Rd, K Logical CM Register and Constant Rd - FM - KR ZON VI 1 CR Rd, K Logical CM Register and Constant Rd - FM - KR ZON VI 1 CR Rd, K Logical CM Register and Constant Rd - FM - KR ZON VI 1 CR Rd, K Logical CM Register Rd - FM - KR ZON VI 1 CR Rd, K Logical	ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1	
SNB Pic. M. Pr. Stores for head press PicM. Pr. ZON /H 1 SNB BA K. M. Subard Construt Register PicM. Pr. C. ZON /H 1 SND RL, K. Subard Construt Registers PicM. Pr. C. ZON /H 1 SNN RL, K. Subard Int Construct Registers PicM. Pr. C. ZON /H 1 SNN RL, K. Subard Int Construct Registers PicM. Pr. Apr. At . ZON /H 1 SNN RL, K. Logica AND Register and Constant PicM. Pr. Apr. At . ZON /H 1 AND RL, K. Logica AND Register and Constant PicM. Pr. Apr. At . ZON /H 1 AND RL, K. Logica AND Register and Constant PicM. Pr. Apr. At . ZON /H 1 COM RL A Exation Pic. Provide State Pic. Pic. Pic. Pic. Pic. Pic. Pic. Pic.	ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2	
SLBI Bask Statemer wice rung we Register Pat - Ba - Nr - G Z.C.N.V.H 1 SBC BB, Br Statemer wice rung we Register Pat - Ba - Nr - G Z.C.N.V.H 1 SBT Rd, K Statemer wice rung we Register Pat - Ra - Nr - G Z.C.N.V.H 1 SBT Rd, K Statemer wite rung we Register Pat - Ra + Nr Z.C.N.V.H 1 AND Rd, K Logical ADR Register ad Contain Pat - Ra + Nr Z.N.V 1 CR Rd, K Logical ADR Register ad Contain Pat - Ra + Nr Z.N.V 1 CR Rd, K Logical OR Register ad Contain Pat - Ra + Nr Z.N.V 1 CR Rd, K Statemer Pat - Ra + Nr Z.N.V 1 CR Rd, K Statemer Pat - Ra + Nr Z.N.V 1 Statemer Rd, K Statemer Pat - Ra + Nr Z.N.V 1 Statemer Rd, K Statemer Pat - Ra + Nr Z.N.V 1 Statemer Rd, K S	SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1	
SBC Bs, R. Subman, Win Carry Non Registers. Piels, Piel, R.C. ZCAN,VH 1 SBCU Bs, K. Subman, Win Carry Constant from Registers. Piels, Piels, C.C. ZCAN,VS 2 AND Bs, R.R. Logial AND Registers. Piels, Piels ZAN,V 1 AND Bs, R.R. Logial AND Register and Constant. Piels, Piels K.C. ZAN,V 1 ORI Bs, R.R. Logial AND Register and Constant. Piels, Piels K.C. ZAN,V 1 ORI Bs, R.K. Logial AND Register and Constant. Piels, Piels, Piels. ZAN,V 1 ORI Bs, R.M. Logial ON Register and Constant. Piels, Piels, Piels. ZAN,V 1 COM RS Data Complement. Piels, Piels, Piels, Piels. ZAN,V 1 COM RS Data Consplement. Piels, Piel	SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1	
SRC Bathmart ImmoSantor Northorn PRH-Bit - RANGH - K Z.C.N.V.H 1 AND PB,K Logial AND Registers PB-Hat - RANGH - K Z.N.V 1 AND PB,K Logial AND Registers and Constant PB - HA - K Z.N.V 1 ORI PB,K Logial ON Registers and Constant PB - HA - K Z.N.V 1 ORI PB,K Logial ON Registers and Constant PB - HA - K Z.N.V 1 COM PB,K Logial ON Register and Constant PB - HA - K Z.N.V 1 EGR PB,K Cogial ON Register and Constant PB - HA - K Z.N.V 1 SBR RAK Statility in Register PB - HA - IA Z.N.V 1 CBR RAK Classifier And Mark PB - HA - IA Z.N.V 1 DBG RAK Classifier And Mark PB - HA - IA Z.N.V 1 DBG RAK Classifier And Mark PA - HA - IA Z.N.V 1 DBG RAK Classifier And IA Z.N	SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1	
BBW Rab. Bubman Irom Monito Ref. Ref. C2.N.V.S 1 AND R6, Rr Logical AND Register and Constant. Rd - R6 v RC Z.N.V I AND R6, Rr Logical AND Register and Constant. Rd - R6 v RC Z.N.V I ORI R5, Rr Logical OR Register and Constant. Rd - R6 v RC Z.N.V I ORI R5, Rr Logical OR Register and Constant. Rd - R6 v RC Z.N.V I COM Rd Onte Complement Rd - R6 v RC Z.N.V I COM Rd Onte Complement Rd - R6 v RC Z.N.V I COM Rd Onte Complement Rd - R6 v RC Z.N.V I SBR Rd R Set Register Rd - R6 v RC Z.N.V I Rd Rd R Set Register Rd - R6 v RC Z.N.V I Rd Rd S Rd R Rd Register Rd Register Z.N.V I Rd Rd S Rd Register Rd Register <t< td=""><td>SBCI</td><td>Rd, K</td><td>Subtract with Carry Constant from Reg.</td><td>$Rd \gets Rd - K - C$</td><td>Z,C,N,V,H</td><td>1</td></t<>	SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \gets Rd - K - C$	Z,C,N,V,H	1	
AND Pis,K Logical AND Pagisters and Constant Pid + Pie + Fr Z.N.V 1 OR Pis,K Logical OR Pagisters and Constant Pid + Pie + N Z.N.V 1 OR Pis,K Logical OR Pagisters and Constant Pid + Pie + N Z.N.V 1 EOR Pis,K Logical OR Pagisters and Constant Pid + Pie + N Z.N.V 1 EOR Pis,K Logical OR Pagisters and Constant Pid + Pid + N Z.N.V 1 EOR Pis Exclusive OR Pagisters Pid + Pid + Pid Z.N.V 1 NEG Pid Two's Complement Pid + Pid + Pid Z.N.V 1 CRI Rick Ges Bittigin Register Pid + Pid + Pid Z.N.V 1 DEG Pid Decomment Pid + Pid + Pid Z.N.V 1 DEG Pid Decomment Pid + Pid + Pid Z.N.V 1 DEG Pid Decomment Pid + Pid + Pid Z.N.V 1 DEG Pid + Pid + Pid + Pid Pid + Pid + Pid	SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2	
AND Rp. R Logad AND Register and Constant Ret - Ret VR Z.N.V 1 OR Rg. R Logad OR Register and Constant Ret - Ret VR Z.N.V 1 OR Rg. R Exclusive OR Registers Ret - Ret VR Z.N.V 1 COM Rd On's Constant Ret - Out" Z.N.V 1 COM Rd On's Constant Ret - Out" Z.N.V 1 COM Rd On's Constant Ret - Ret VR Z.N.V 1 SR Rd Set Register Ret - Ret VR Z.N.V 1 DEC Rd Incomment Ret - Ret 1 Z.N.V 1 INC Rd Test D/ Zaro or Minia Ret - Ret 1 Z.N.V 1 SR Rd Test D/ Zaro or Minia Ret - Ret 1 Z.N.V 1 SR Rd Ret Ret 3 Rd Ret 3 Z.N.V 1 SR Rd Ret 3 Ret Ret 3 Z.N.V 1 Z.N.V 1 SR	AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1	
OR Rg,K Logard OR Register an Constant Rd + Rd v Fr Z,N.V 1 DGR Rg,K Logard OR Register an Constant Rd + Rd v K Z,N.V 1 DGR Rg,R Exclusive CIR Register Rd + Rd v K Z,N.V 1 DGR Rd Tors Complement Rd + Ox0 - Rd Z,O.N.V 1 DSR Rd X Ste Bill() in Register Rd + Rd + Ox0 - Rd Z,N.V 1 DGR Rd X Clear Ring Register Rd + Rd + Rd + OxFF + Rd Z,N.V 1 DSR Rd X Decomment Rd + Rd + Rd + Rd + Rd + Rd Z,N.V 1 DSR Rd Decomment Rd + Rd + Rd + Rd Z,N.V 1 DSR Rd Ste Register Rd + Rd + Rd + Rd Z,N.V 1 DLL Rd Decomment Rd + Rd	ANDI	Rd, K	Logical AND Register and Constant	$Rd \gets Rd \bullet K$	Z,N,V	1	
OH Hg, R Logae OF Register and Constant He / Ho v / Ho / Ho / Ho / Constant Ret / Ho / Ho / Constant Ret / Ho / Sec Z,N,V 1 COM Hg Ores Complement Hd + Ouf? - Fid Z,N,V 1 SBR Hg/K SetBiolin Register Hd + Ouf? - Fid Z,N,VI 1 SBR Rg/K SetBiolin Register Hd + Rd +	OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \lor Rr$	Z,N,V	1	
EOR. Rd, Pr Exclusion OR Registers Pd + - Rd + GPF - Pd Z.N.V 1 OCM Rd Two's Complement Pd + - 0.00 - Pd Z.N.V 1 SBR RdX Set Biglin In Register Pd + - 0.00 - Pd Z.N.V 1 DBR RdX Clear Biglin Register Pd + - 0.00 - Pd Z.N.V 1 DBR RdX Clear Biglin Register Pd + - Rd + 0.02 F - N; Z.N.V 1 DBC Rd Decement Pd + - Rd + 1.02 - Rd Z.N.V 1 DCR Rd Deservent Pd + - Rd + Rd - 1 Z.N.V 1 DCR Rd Deservent Pd + - Rd + Rd Z.N.V 1 DCR Rd Deservent Pd + - Rd + Rd Z.N.V 1 DCR Rd Deservent Pd + - Rd + Rd Z.N.V 1 DCR Rd Deservent Pd + - Rd + Rd Z.N.V 1 DCR Rd Pd + Pd + Rd + Rd Pd + - Rd + Rd Z.N.V 1	ORI	Rd, K	Logical OR Register and Constant	$Rd \gets Rd \lor K$	Z,N,V	1	
COM R4 One's Complement P4 - 0.6F - P4 Z.C.N.V 1 NBE0 R4 Nova Complement R4 - R00 - P4 Z.C.N.VI 1 S8R R4/K Statu) in Register R4 - R4 v K Z.N.VI 1 S0R R4/K Statu) in Register R4 - R4 v K Z.N.VI 1 INC R4 Decrement R4 - R4 v Agr - R4 v Agr - R4 Z.N.VI 1 INC R4 Decrement R4 - R4 v Agr - R4 v Agr - R4 Z.N.VI 1 IST R4 Caser Register R4 - R4 v Bd Z.N.VI 1 SER R4 Caser Register R4 - R4 v R Z.C.VI 2 VAUS R4 Rr Multyly Unsigned R1R0 - R4 v Rr Z.C. 2 VAUS R4 Rr Multyly Unsigned R1R0 - R4 v Rr Z.C. 2 VAUS R4 Rr Fractional Multyly Unsigned R1R0 - (Max R) v<1	EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1	
NEG Rd K Star Blug) in Register Rd 0.00 - Rd Z.C.N.V.H 1 CBR Rd K Construinty in Register Rd Rd v K Z.N.V 1 CBR Rd K Construinty in Register Rd +- Rd + 0.07F + N,O Z.N.V 1 DEG Rd Decrement Rd +- Rd + 0.07F + N,O Z.N.V 1 DEG Rd Destroment Rd +- Rd + Rd Z.N.V 1 CLR Rd Destroment Rd +- Rd + Rd Z.N.V 1 CLR Rd Destroment Rd +- Rd + Rd Z.N.V 1 CLR Rd Destroment Rd +- Rd + Rd Z.N.V 1 SER Rd Destroment Rd +- Rd + Rd Z.C. 2 MULS Rd, Rr Hallely bingend R1:Rd + Rd + Rd Z.C. 2 MULSU Rd, Rr Factorian Multiply Signed with Unsigned R1:Rd + Rd x Rd Z.C. 2 MULSU Rd, Rr Factorian Multiply Signed with Unsigned R1:Rd + Rd x Rd Z.N	COM	Rd	One's Complement	$Rd \leftarrow 0xFF - Rd$	Z,C,N,V	1	
BBR Pid, K Set Bit(s) in Register Pid - Rd v K Z.N.V 1 CBR RdX Clear Bit(s) in Register Pid - Rd v (DKF - K) Z.N.V 1 INC Rd Decomment Rd - Rd v 1.0 Z.N.V 1 IST Rd Decomment Rd - Rd v 1.0 Z.N.V 1 IST Rd Clear Register Rd - Rd v Rd Z.N.V 1 IST Rd Clear Register Rd - Rd v Rd Z.N.V 1 SER Rd R Multyly Unogvad Rd rd v Rd	NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1	
CBR Rdx Class Bitg) in Register Rd - RH - RH - (MCF + K), Month Z.N.V 1 DEC Rd Decomment Rd - RH - 1 Z.N.V 1 DEC Rd Decomment Rd - RH - 1 Z.N.V 1 DEC Rd Test to Zen or Minus Rd - Rd - Rd Z.N.V 1 CLR Rd Clear Register Rd - Alg - Rd None 1 MLR Rd, Rr Multpy Lingged RH - PG - Rd x Rr Z.C 2 MULS Rd, Rr Multpy Signed with Unsigned RH 190 - Rd x Rr Z.C 2 2 RMLS Rd, Rr Pactional Multpy Signed with Unsigned RH 190 - Rd x Rr < <td>Z.C 2 2 RMLS Rd, Rr Pactional Multpy Signed with Unsigned RH 190 - (Rd x Rr) <<1</td> Z.C 2 2 RMLS Rd, Rr Pactional Multpy Signed with Unsigned RH 190 - (Rd x Rr) <<1	Z.C 2 2 RMLS Rd, Rr Pactional Multpy Signed with Unsigned RH 190 - (Rd x Rr) <<1	SBR	Rd,K	Set Bit(s) in Register	$Rd \gets Rd \lor K$	Z,N,V	1
INC Pid Interment Pid - RH - 1 ZNV 1 DEC Rd Decrement Rd - RH - 1 ZNV 1 TST Rd Test for Zero or Muus Rd - RH - Rd - Rd ZNV 1 CIR Rd Char Register Rd - Ad + Rd ZNV 1 EER Rd Set Register Rd - Ad + Rd ZCC 2 MULS Rd, Rr Multyly Unsigned R1 R0 - Rd x Rr ZCC 2 MULS Rd, Rr Fractional Multyly Unsigned R1 R0 - (Rd x Rr) Z.C 2 MULS Rd, Rr Fractional Multyly Unsigned R1 R0 - (Rd x R) <<1	CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (0xFF - K)$	Z,N,V	1	
DECPdDecrementPd - Pd - 1Z,N V1TSTRdTestor zor MinusRd - Pd r RdZ,N V1CLRPd dClear RegisterRd - Pd r RdZ,N V1SERRdSet RegisterRd - OdFNone1MULRd, RrMultiply UnsignedPd r Ads RrZ,C2MULSRd, RrMultiply SignedPd r Ads RrZ,C2MULSRd, RrMultiply Signed with UnsignedPd r Ads RrZ,C2PMULRd, RrFractional Multiply SignedPd r Ads RrZ,C2PMULSRd, RrFractional Multiply SignedPd r Ads RrZ,C2PMULSRd, RrFractional Multiply SignedPd r Ads Rr N r Ads RrZ,C2PMULSRd, RrFractional Multiply Signed with UnsignedPd r Ads Rr N r Ads RrZ,C2PMULSRd, RrFractional Multiply Signed with UnsignedPd r Ads Rr N r Ads RrZ,C2PMULSRd, RrFractional Multiply Signed with UnsignedPd r Ads Rr N r Ads RrZ,C2IMPkPd endational Multiply Signed with UnsignedPd r Ads Rr N r Ads RrZ,C2IMPkPd endational Multiply Signed with UnsignedPd r Ads RrX,C2IMPkPd endational Multiply Signed with UnsignedPd r Ads RrX,C2IMPkPd endational Multiply Signed with UnsignedPd r Ads RrX,C2IMARKPd en	INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1	
TSTPdTest for Zero or MinusPd \leftarrow Pd \leftarrow Pd \leftarrow Z,NV1CLRRdClear RegisterPd \leftarrow Pd \leftarrow Pd \leftarrow Z,NV1SERPd \leftarrow Pd \leftarrow None1MULPd, PrMuliply UnsignedPd \leftarrow Pd \leftarrow Z,C2MULSPd, PrMuliply UnsignedPd $+$ Pd $+$ Z,C2MULSUPd, PrMuliply SignedPd $+$ Pd $+$ Z,C2MULSUPd, PrMuliply SignedPd $+$ Pd $+$ Z,C2PMULRd, PrFractional Muliply SignedPd $+$ Pd $+$ Z,C2PMULSUPd, PrFractional Muliply SignedPd $+$ Pd $+$ Z,C2PMULSUPd, PrFractional Muliply SignedPd $+$ Pd $+$ Z,C2PMULSUPd, PrFractional Muliply SignedPd $+$ Pd $+$ Z,C2PMUSUPd $+$ Pd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2PMUSUPd $+$ Pd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2Signed Pd $+$ Pd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2PMUSUPd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2PMUSUPd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2Signed Pd $+$ Pd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2Signed Pd $+$ Pd $+$ Pd $+$ Pd $+$ Pd $+$ Z,C2<	DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1	
CLRPatDate ArabitCarrowNone1SERRdSat RegisterPat - 0.0FNone1MULPat, RrMuliply UnsignedPat Pat S PrZ.C2MULSUPat, RrMuliply Signed with UnsignedPat Pat S PrZ.C2MULSUPat, RrMuliply Signed with UnsignedPat Pat S PrZ.C2PMULRd, RrFractional Multiply Signed with UnsignedPat Pat - Reid S PriceZ.C2PMULSURd, RrFractional Multiply Signed with UnsignedPat Pat - Reid S PriceZ.C2PMULSURd, RrFractional Multiply Signed with UnsignedPat Pat - Reid S PriceZ.C2PMULSURd, RrFractional Multiply Signed with UnsignedPat Pat - Reid S PriceZ.C2PMACH MSTRUCTONSPat Pat S PriceNone2IMPkReistree JungPC + PC + k + 1None3RCALLkIndext Call to (2)PC + ZNone3CALLkDirect Subroutine CallPC + STACKNone4RETSubroutine ReturnPC - STACKNone4RETSubroutine ReturnPC - STACKNone4CPSRd, RrCompare Signi EqualIf (Rd - R) PC + PC + 2 or 3None11/23CPCRd, RrCompare Signi EqualIf (Rd - R) PC + PC + 2 or 3None11/23CPLRd, RrCompare Signi EqualIf (Rd - R) PC + C + C + C + 3None <td>TST</td> <td>Rd</td> <td>Test for Zero or Minus</td> <td>$Rd \leftarrow Rd \bullet Rd$</td> <td>Z,N,V</td> <td>1</td>	TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1	
SER Rd Rd Set Pogletr Rd Rd $None$ 1MULBd, RrMultgy Signed $R1:B0 \leftarrow Rd \times Rr$ Z.C2MULSUBd, RrMultgy Signed $R1:B0 \leftarrow Rd \times Rr$ Z.C2MULSURd, RrFactional Multgy Signed $R1:B0 \leftarrow Rd \times Rr$ Z.C2FMULSRd, RrFractional Multgy Signed $R1:B0 \leftarrow Rd \times Rr <<1$ Z.C2FMULSURd, RrFractional Multgy Signed $R1:B0 \leftarrow (Rd \times Rr) <<1$ Z.C2FMULSURd, RrFractional Multgy Signed $R1:R0 \leftarrow (Rd \times Rr) <<1$ Z.C2FMULSURd, RrFractional Multgy Signed $R1:R0 \leftarrow (Rd \times Rr) <<1$ Z.C2BRANCH INSTRUCTIONSFFFNone2IMPkRelative JumpPC \leftarrow PC + k + 1None3ICALLkRelative Surroutine CallPC \leftarrow RNone3ICALLkDirect Jump In (Z)PC \leftarrow TNone4ICALLkDirect Jump In (Z)PC \leftarrow TNone4ICALLkDirect Jump In (Z)PC \leftarrow TNone3ICALLkDirect Jump In (Z)PC \leftarrow TNone4ICALLkDirect Jump In (Z)PC \leftarrow TNone1	CLR	Rd	Clear Register	$Rd \ \leftarrow Rd \oplus Rd$	Z,N,V	1	
MULR1, RrMultipy UnsignedR1:R0 - Rd x FrZ.C2MULSURd, RrMultipy SignedR1:R0 - Rd x FrZ.C2MULSURd, RrPractional Multiply UnsignedR1:R0 - Rd x FrZ.C2FMULRd, RrPractional Multiply Signed with UnsignedR1:R0 - (Rd x R) <<1	SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1	
MULSRd, RrMultiply Signed with UnsignedR1:R0 - Rd x RrZ.C2FMULSRd, RrFractional Multiply Signed with UnsignedR1:R0 - Rd x RrZ.C2FMULSRd, RrFractional Multiply Signed with UnsignedR1:R0 - (Rd x R) <<1	MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2	
MULSURd, RrMultiply Signed with UnsignedR1:R0 - (Rd x Rr)Z.C2FMULRd, RrFractional Multiply SignedR1:R0 - (Rd x Rr) <1	MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2	
FMUL Pic, Pr Fractional Multiply Unsigned R1:80 - (Rd x R) ≪1 Z,C 2 FMULS Rd, Rr Fractional Multiply Signed with Unsigned R1:80 - (Rd x R) ≪1 Z,C 2 BRANCH INSTRUCTIONS Rd, Rr Fractional Multiply Signed with Unsigned R1:80 - (Rd x R) ≪1 Z,C 2 BMMP k Relative Jump PC + PC + k + 1 None 2 JMP k Direct Jump ID (2) PC + C + k None 3 ICALL k Relative Subroutine Call PC + PC + k + 1 None 3 ICALL k Direct Subroutine Call PC + Z None 4 RET Subroutine Relum PC + STACK None 4 RET Subroutine Relum PC + STACK None 1/2/3 CP Rd,Rr Compare Rd - Rr Z, NV,C,H 1 CPC Rd,Rr Compare Register Vieth Immediate Rd - K Z, NV,C,H 1 CP Rd,Rr Compare Register Vieth Immediate Rd	MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd x Rr$	Z,C	2	
IFMULSUPickonal Multiply SignedPickonal Multiply Signed with UnsignedPickonal Multiply Signed With Signed	FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2	
FMULSUPractional Multiply Signed with UnsignedPLR0 - (Rd x Rr) << 1Z, C2BRANCH INSTRUCTIONSFUMPkRelative JumpPC - PC + k + 1None2JMPindirect Jump to (2)PC - ZNone3ICALLkDirect JumpPC - KNone3ICALLkRelative Subroutine CallPC - FC + k + 1None3ICALLkDirect Subroutine CallPC - FC + k + 1None4ETTSubroutine RelumPC - STACKNone4RETTSubroutine RelumPC - STACKNone4CPSERd,RrCompare, Skip if Equalif (Rd = Rr) PC - FC + 2 or 3None11/2/3CPRd,RrCompare with CarryRd - Rr - CZ, N, V, C, H1CPCRd,RrCompare Relister with ImmediateRd - KZ, N, V, C, H1CPCRd,RrCompare Relister with ImmediateRd - KZ, N, V, C, H1CPCRd,RrSkip if Bit in Register is Setif (Rt(b)=1) PC - PC + 2 or 3None1/2/3SBRCP, bSkip if Bit in Register is Setif (Rt(b)=1) PC - PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register is Setif (Rt(b)=1) PC - PC + 2 or 3None1/2/3SBRSP, bSkip if Bit in Register is Setif (Rt(b)=1) PC - PC + 2 or 3None1/2/3SBRSP, bSkip if Bit in Register is Setif (Rt(b)=1) PC - PC + 2 or 3None1/2/3 <td< td=""><td>FMULS</td><td>Rd, Rr</td><td>Fractional Multiply Signed</td><td>R1:R0 ← (Rd x Rr) << 1</td><td>Z,C</td><td>2</td></td<>	FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2	
BRANCH INSTRUCTIONS BLMP k Relative Jump PC + PC + k + 1 None 2 JMP k Direct Jump PC + K None 3 GCALL k Relative Subroutine Call PC - PC + k + 1 None 3 ICALL k Relative Subroutine Call PC - PC + k + 1 None 3 ICALL k Direct Subroutine Call PC - EC + k + 1 None 4 RET Subroutine Return PC - STACK None 4 RET Interrupt Return PC - STACK In 4 CPSE Rd,Rr Compare, Skip if Equal if (Rd = Rr) PC + PC + 2 or 3 None 1/2/3 CP Rd,Rr Compare Register with immediate Rd - K Z, N,V,C,H 1 SBRC Rr, b Skip if Bit in Register Cleared if (R(b)=1) PC + PC + 2 or 3 None 1/2/3 SBRS P, b Skip if Bit in VG Register Cleared if (P(b)=0) PC + PC + 2 or 3 None 1/2/3 BRS P, b Skip	FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd x Rr) << 1$	Z,C	2	
FLMPkRelative JumpPC PC + k + 1None2JMPindirect Jump to (2)PC CZNone2JMPkDirect JumpPC CZNone3RCALLkRelative Subroutine CallPC PC + k + 1None3CALLLIndirect Call to (2)PC ZNone3CALLkDirect Subroutine CallPC CZNone4RETSubroutine ReturnPC STACKNone4RETInterrupt ReturnPC STACKI4CPSERd,RrCompare, Skip if EqualIf (Rd = Rt) PC + DC + 2 or 3None1/2/3CPRd,RrCompare with CarryRd RtrZ, NV, C, H1CPCRd,RrCompare with CarryRd RtrZ, NV, C, H1CPIRd,XCompare with CarryRd RtrZ, NV, C, H1CPIRd,XCompare Register vith ImmediateRd KZ, NV, C, H1CPIRd,XSkip If Bit In Register ClearedIf (Rtr(b)=1) PC + DC + 2 or 3None1/2/3SBRSRt, bSkip If Bit In I/O Register ClearedIf (Rtr(b)=1) PC + DC + 2 or 3None1/2/3SBRSP, bSkip If Bit In I/O Register ClearedIf (Rtr(b)=1) PC + DC + 2 or 3None1/2/3SBRSP, bSkip If Bit In I/O Register ClearedIf (RtG)=0 then PC + DC + 4 or 3None1/2/3SBRSP, bSkip If Bit In I/O Register ClearedIf (PC + DC + 2 or 3	BRANCH INSTRUCT	TIONS			1		
$ \begin{array}{ c c c c } LMP & h & Direct Jump to (Z) & PC \leftarrow Z & None & 2 \\ \hline MP & h & Direct Jump & PC \leftarrow k & None & 3 \\ \hline RCALL & k & Petative Subroutine Call & PC \leftarrow PC + k + 1 & None & 3 \\ \hline ICALL & L & Indirect Call to (Z) & PC \leftarrow Z & None & 3 \\ \hline ICALL & k & Direct Subroutine Call & PC \leftarrow k & None & 4 \\ \hline RET & Subroutine Call & PC \leftarrow k & None & 4 \\ \hline RET & Subroutine Return & PC \leftarrow STACK & None & 4 \\ \hline RET & Interrupt Return & PC \leftarrow STACK & None & 4 \\ \hline RET & Interrupt Return & PC \leftarrow STACK & None & 1 \\ \hline OPSE & Rd,Rr & Compare, Skip if Equal & If (Rd = Rr) PC \leftarrow PC + 2 or 3 & None & 11/2/3 \\ \hline CPR & Rd,Rr & Compare, Skip if Equal & If (Rd = Rr) PC \leftarrow PC + 2 or 3 & None & 11/2/3 \\ \hline CPR & Rd,Rr & Compare Name Michael & Rd - K & Z, NV,C,H & 1 \\ \hline CPC & Rd,Kr & Compare Name Michael & Rd - K & Z, NV,C,H & 1 \\ \hline SBRC & Rr, b & Skip If Bit In Register Is Set & If (R(b)=0) PC \leftarrow PC + 2 or 3 & None & 11/2/3 \\ \hline SBRS & Rr, b & Skip If Bit In Register Is Set & If (R(b)=0) PC \leftarrow PC + 2 or 3 & None & 11/2/3 \\ \hline SBRS & P, b & Skip If Bit In I/O Register Cleared & If (P(b)=1) PC - PC + 2 or 3 & None & 11/2/3 \\ \hline SBRS & P, b & Skip If Bit In I/O Register Cleared & If (P(b)=0) PC - PC + 2 or 3 & None & 11/2/3 \\ \hline SBRS & P, b & Skip If Bit In I/O Register Cleared & If (P(b)=0) PC - PC + 2 or 3 & None & 11/2/3 \\ \hline SBRS & P, b & Skip If Bit In I/O Register Set & If If (RD(b)=1) PC - PC + 2 or 3 & None & 11/2/3 \\ \hline SBRS & s, k & Branch If Status Flag Set & If ISREG(s) = 0) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & s, k & Branch If Status Flag Set & If ISREG(s) = 0) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & k & Branch If Status Flag Set & If I(SEG(s) = 0) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & k & Branch If Status Flag Set & If I(C = 1) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & k & Branch If Max & If I(C = 0) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & k & Branch If Max & If I(C = 0) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & k & Branch If Max & If I(C = 0) Inten PC - PC + k + 1 & None & 11/2 \\ \hline SRRS & k & Branch I$	RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2	
JMPkDirect JumpPC $\leftarrow k$ None3IGALLkRelative Subroutine CallPC \leftarrow PC \leftarrow k + 1None3IGALLkDirect Subroutine CallPC $\leftarrow L$ None3IGALLkDirect Subroutine CallPC $\leftarrow L$ None4RETSubroutine ReturnPC $\leftarrow STACK$ None4RETInterrupt ReturnPC $\leftarrow STACK$ I4CPSERd,RrCompare, Skip if Equalif (Rd = R) PC $\leftarrow PC + 2 \text{ or 3}$ None1/2/3CPRd,RrCompare Register with ImmediateRd $-$ Rr $-$ CZ, N, V, C, H1CP1Rd,KCompare Register with Immediateif (Rf(D)=0) PC $\leftarrow PC + 2 \text{ or 3}$ None1/2/3SBRSRr, bSkip if Bit in Register is Setif (Rr(D)=1) PC $\leftarrow PC + 2 \text{ or 3}$ None1/2/3SBRSRr, bSkip if Bit in Register is Setif (Rr(D)=1) PC $\leftarrow PC + 2 \text{ or 3}$ None1/2/3SBRSP, bSkip if Bit in VO Register Clearedif (P(b)=0) PC $\leftarrow PC + 2 \text{ or 3}$ None1/2/3SBRSS, kBranch if Status Flag Setif (RSEG(s)=1) fhen PC $\leftarrow PC + k + 1$ None1/2BRBCs, kBranch if Status Flag Clearedif (Z = 0) fhen PC $\leftarrow PC + k + 1$ None1/2BRBCs, kBranch if Status Flag Clearedif (Z = 0) fhen PC $\leftarrow PC + k + 1$ None1/2BRBCkBranch if Carry Setif (C = 0) fhen PC $\leftarrow PC + k + 1$ None1/2BRACkBranch if Carr	IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2	
RCALLkRelative Subroutine CallPC \leftarrow PC + k + 1None3ICALLindirect Call to [2]PC \leftarrow ZNone3CALLkDirect Subroutine CallPC \leftarrow KNone4RETSubroutine ReturnPC \leftarrow STACKNone4RETInterrupt ReturnPC \leftarrow STACKI4CPSERd,RrCompare, Skip if Equalif (Rd = Rr) PC \leftarrow PC + 2 or 3None11/2/3CPRd,RrCompare with CarryRd $-$ Rr $-$ CZ, NV,C,H1CPCRd,RrCompare with CarryRd $-$ Rr $-$ CZ, NV,C,H1CPIRd,KCompare Register with ImmediateRd $-$ KZ, NV,C,H1SBRSRr, bSkip If Bit in Register Clearedif (Pr(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip If Bit in Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBICP, bSkip If Bit in Ogeister Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip If Bit in UO Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip If Bit in UO Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip If Bit in UO Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISS, kBranch if Satus Flag Setif (C R) In PC \leftarrow PC + K + 1None1/2BRBCs, kBranch if Satus Flag Clearedif (P(b)	JMP	k	Direct Jump	$PC \leftarrow k$	None	3	
ICALLIndirect Call to (Z)PC \leftarrow ZNone3CALLkDirect Subroutine CallPC \leftarrow STACKNone4RETSubroutine ReturnPC \leftarrow STACKI4RETInterrupt ReturnPC \leftarrow STACKI4CPSERd,RrCompare, Skip if Equalif (Rd = Rr) PC \leftarrow PC + 2 or 3None11/2/3CPRd,RrCompare with CarryRd $-$ Rr $-$ CZ, N, V, C, H1CPCRd,RrCompare with CarryRd $-$ Rr $-$ CZ, N, V, C, H1CP1Rd,KCompare Register with ImmediateRd $-$ KrZ, N, V, C, H1CP2Skip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None11/2/3SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=1) PC \leftarrow PC + 2 or 3None11/2/3SBRSRr, bSkip if Bit in UO Register Setif (P(b)=1) PC \leftarrow PC + 2 or 3None11/2/3SBRSP, bSkip if Bit in UO Register Setif (P(b)=1) PC \leftarrow PC + 2 or 3None11/2/3SBRSP, bSkip if Bit in UO Register Setif (P(b)=1) PC \leftarrow PC + 2 or 3None11/2/3BRSs, kBranch if Status Flag Setif (P(b)=1) PC \leftarrow PC + 2 or 3None11/2/3BRBCs, kBranch if Status Flag Setif (P(b)=1) PC \leftarrow PC + C + C + 1None11/2BRBCs, kBranch if Not Equalif (Z = 1) then PC \leftarrow PC + k + 1None11/2BRNEkBranch if Not Equalif (C = 0)	RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3	
CALLkDirect Subroutine CallPC \leftarrow StackNone4RETSubroutine ReturnPC \leftarrow STACKNone4RET1Interrupt ReturnPC \leftarrow STACKI4CPSERd,RrCompare, Skip if Equalif (Rd = R) PC \leftarrow PC + 2 or 3None1/2/3CPRd,RrCompareRd - Rr - CZ, N, V, C, H1CPCRd,RrCompare Null CarryRd - Rr - CZ, N, V, C, H1CPIRd,KCompare Register with ImmediateRd - KZ, N, V, C, H1SBRCRr, bSkip if Bt in Register I Setif (Rt(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRCP, bSkip if Bt in Register I Setif (Rt(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBRCP, bSkip if Bt in In QRegister Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSP, bSkip if Bt in I/O Register is Setif (Rt(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBRSs, kBranch if Status Flag Setif (RtGls(s) = 1) then PC \leftarrow PC + k + 1None1/2BRBSs, kBranch if Status Flag Clearedif (REGls(s) = 0) then PC \leftarrow PC + k + 1None1/2BREOkBranch if Status Flag Clearedif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRECkBranch if Carry Clearedif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRECkBranch if Carry Clearedif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRECk <td>ICALL</td> <td></td> <td>Indirect Call to (Z)</td> <td>$PC \leftarrow Z$</td> <td>None</td> <td>3</td>	ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3	
RETSubroutine ReturnPC \leftarrow STACKNone4RETIInterrupt ReturnPC \leftarrow STACKI4CPSERd,RrCompare, Skip if Equalif (Rd = Rr) PC \leftarrow PC + 2 or 3None1/2/3CPRd,RrCompare, Skip if Equalif (Rd = Rr) PC \leftarrow PC + 2 or 3None1/2/3CPCRd,RrCompare with CarryRd - RrZ, N,V,C,H1CPCRd,KCompare Register with ImmediateRd - KZ, N,V,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register Clearedif (R(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSP, bSkip if Bit in NO Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3BRSs, kBranch if Status Flag Setif (Re)(S)=0) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Setif (SREG(s)=0) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Gatus Flag Clearedif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Garey Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Garey Clearedif (N = 0) then PC \leftarrow PC + k + 1None <td< td=""><td>CALL</td><td>k</td><td>Direct Subroutine Call</td><td>$PC \leftarrow k$</td><td>None</td><td>4</td></td<>	CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	4	
RETIInterrupt ReturnPC <-STACKI4CPSERd,RrCompare, Skip if Equalif (Rd = Rr) PC <-PC + 2 or 3	RET		Subroutine Return	$PC \leftarrow STACK$	None	4	
CPSERd,RrCompare, Skip if Equalif (Rd = Rr) PC \leftarrow PC + 2 or 3None1/2/3CPRd,RrCompareCompare with CarryRd - RrZ, N,V,C,H1CPCRd,RrCompare with CarryRd - Rr - CZ, N,V,C,H1CPIRd,KCompare Register with ImmediateRd - Rr - CZ, N,V,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in log Register Clearedif (Rr(b)=1) PC ← PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register Clearedif (P(b)=0) PC ← PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register is Setif (SREG(s) = 1) then PC ← PC + k + 1None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 0) then PC ← PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (Z = 0) then PC ← PC + k + 1None1/2BREQkBranch if Not Equalif (Z = 0) then PC ← PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC ← PC + k + 1None1/2BRCCkBranch if Carry Glearedif (C = 0) then PC ← PC + k + 1None1/2BRCSkBranch if Carry Glearedif (C = 0) then PC ← PC + k + 1None1/2BRCSkBranch if Lawer OFif (C = 0) then PC ← PC + k + 1None1/2BRCSkBranch if Minusif (RETI		Interrupt Return	$PC \leftarrow STACK$	1	4	
CPRd,RrCompareRd - RrZ, NV,C,H1CPCRd,RrCompare with CarryRd - Rr - CZ, NV,C,H1CPIRd,KCompare Register with ImmediateRd - KZ, NV,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register Clearedif (Rr(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in I/O Register Clearedif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register Clearedif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (Z = 1) then PC \leftarrow PC + k + 1None1/2BRBCkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Carry Clearedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHkBranch if Greater or Equal, Signedif	CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC \leftarrow PC + 2 or 3	None	1/2/3	
CPCRd, RrCompare with CarryRd - Rr - CZ, N, V, C, H1CPIRd, KCompare Register with ImmediateRd - KZ, N, V, C, H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register is Setif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBICP, bSkip if Bit in I/O Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register Setif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC+-PC+k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC+-PC+k + 1None1/2BREQkBranch if Status Flag Clearedif (Z = 0) then PC+-PC+k + 1None1/2BRNEkBranch if Carry Setif (C = 0) then PC+PC+k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC+PC+k + 1None1/2BRNEkBranch if Carry Clearedif (C = 0) then PC+PC+k + 1None1/2BRSHkBranch if Carry Clearedif (C = 0) then PC+PC+k + 1None1/2BRHNkBranch if Garry Clearedif (N = 0) then PC + PC + k + 1None1/2BRSHkBranch if Garry Clearedif (N = 0) then PC + PC + k + 1None1/2BRHNkBranch if Malf Carry Flag Set	CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1	
CPIHd,KCompare Hegister with ImmediateHd - KZ, N,V,C,H1SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register Clearedif (Rr(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBICP, bSkip if Bit in I/O Register Clearedif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register Clearedif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (Z = 1) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Game or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Game or Higherif (N = 1) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Game or Figureif (N = 1) then PC \leftarrow PC + k + 1None1/2BRHkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHkBranch	CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1	
SBRCRr, bSkip if Bit in Register Clearedif (Rr(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBRSRr, bSkip if Bit in Register is Setif (Rr(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBICP, bSkip if Bit in I/O Register Clearedif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register Setif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Status Flag Clearedif (Z = 1) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Oxt Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Setif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLUkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRLUkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHLKBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRHSkBr	CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1	
SBRSHr, bSkip if Bit in Register is Setif (Hr(b)=1) PC \leftarrow PC + 2 or 3None1/2/3SBICP, bSkip if Bit in I/O Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register is Setif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRBCs, kBranch if Status Flag Setif (SREG(s) = 0) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Attas Flag Clearedif (Z = 1) then PC \leftarrow PC + k + 1None1/2BREQkBranch if Carry Setif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCDkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Lawerif (C = 0) then PC \leftarrow PC + k + 1None1/2BRCLkBranch if Musif (N = 1) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Musif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Lasry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHSkB	SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC \leftarrow PC + 2 or 3	None	1/2/3	
SBICP, bSkip if Bit in I/O Register Clearedif (P(b)=0) PC \leftarrow PC + 2 or 3None1/2/3SBISP, bSkip if Bit in I/O Register is Setif (P(b)=1) PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif (SREG(s) = 1) then PC ← PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif (SREG(s) = 0) then PC ← PC + k + 1None1/2BREQkBranch if Equalif (Z = 1) then PC ← PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC ← PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC ← PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC ← PC + k + 1None1/2BRCDkBranch if Carry Clearedif (C = 0) then PC ← PC + k + 1None1/2BRCDkBranch if Carry Clearedif (C = 0) then PC ← PC + k + 1None1/2BRCDkBranch if Minusif (N = 0) then PC ← PC + k + 1None1/2BRHNkBranch if Lowerif (N = 0) then PC ← PC + k + 1None1/2BRPLkBranch if Greater or Equal, Signedif (N = 0) then PC ← PC + k + 1None1/2BRLTkBranch if Half Carry Flag Setif (N = 0) then PC ← PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 0) then PC ← PC + k + 1None1/2BRHCkBranch if Half Carry Fl	SBRS	Rr, b	Skip if Bit in Register is Set	if $(\text{Rr}(b)=1) \text{PC} \leftarrow \text{PC} + 2 \text{ or } 3$	None	1/2/3	
SBISP, DSktp If Bit in 1/O Hegister IS SetIf $(P(b)=1)$ PC \leftarrow PC + 2 or 3None1/2/3BRBSs, kBranch if Status Flag Setif $(SREG(s) = 1)$ then PC \leftarrow PC + k + 1None1/2BRBCs, kBranch if Status Flag Clearedif $(SREG(s) = 0)$ then PC \leftarrow PC + k + 1None1/2BRBQkBranch if Equalif $(Z = 1)$ then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Not Equalif $(Z = 0)$ then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif $(C = 0)$ then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif $(C = 0)$ then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif $(C = 0)$ then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif $(C = 1)$ then PC \leftarrow PC + k + 1None1/2BRHIkBranch if Minusif $(N = 1)$ then PC \leftarrow PC + k + 1None1/2BRLDkBranch if Greater or Equal, Signedif $(N \oplus V = 0)$ then PC \leftarrow PC + k + 1None1/2BRLkBranch if Greater or Equal, Signedif $(N \oplus V = 1)$ then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Larry Flag Setif $(H = 1)$ then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif $(H = 1)$ then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half Carry Flag Setif $(H = 0)$ then PC \leftarrow PC + k + 1None1/2 </td <td>SBIC</td> <td>P, D</td> <td>Skip if Bit in I/O Register Cleared</td> <td>If $(P(b)=0) PC \leftarrow PC + 2 \text{ or } 3$</td> <td>None</td> <td>1/2/3</td>	SBIC	P, D	Skip if Bit in I/O Register Cleared	If $(P(b)=0) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3	
BHBSS, KBranch if Status Flag SetIf (SHEG(S) = 1) then PC-C+C+K + 1None1/2BRBCS, KBranch if Status Flag Clearedif (SREG(S) = 0) then PC-C+C+K + 1None1/2BREQkBranch if Equalif (Z = 1) then PC - PC + k + 1None1/2BRNEkBranch if Not Equalif (Z = 0) then PC - PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC - PC + k + 1None1/2BRCCkBranch if Carry Setif (C = 0) then PC - PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC - PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC - PC + k + 1None1/2BRUkBranch if Greater or Equal, Signedif (N = 1) then PC - PC + k + 1None1/2BRELkBranch if Greater or Equal, Signedif (N = 0) then PC - PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC - PC + k + 1None1/2BRHSkBranch if Half Carry Flag Clearedif (N \oplus V = 0) then PC - PC + k + 1None1/2BRHSkBranch if Half Carry Flag Clearedif (N \oplus V = 1) then PC - PC + k + 1None1/2BRHSkBranch if Half Carry Flag Clearedif (H = 0) then PC - PC + k + 1None1/2BRHSkBranch if Half Carry Flag Clearedif (H = 0) then PC - PC + k + 1None1/2BRHS <t< td=""><td>SBIS</td><td>P, D</td><td>Skip if Bit in I/O Register is Set</td><td>If $(P(D)=1) PC \leftarrow PC + 2 \text{ or } 3$</td><td>None</td><td>1/2/3</td></t<>	SBIS	P, D	Skip if Bit in I/O Register is Set	If $(P(D)=1) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3	
brocs, kBranch if Status Hag GlearedIf (SHEG(s) = 0) then PCC+PC+k+1None1/2BREQkBranch if Equalif (Z = 1) then PC \leftarrow PC + k + 1None1/2BRNEkBranch if Not Equalif (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Lowerif (N = 1) then PC \leftarrow PC + k + 1None1/2BRDLkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRELkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRELkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRELkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half Carry Flag Clearedif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if Telar Setif (T = 1) the	BRBS	S, K	Branch if Status Flag Set	If $(SREG(s) = 1)$ then $PC \leftarrow PC + k + 1$	None	1/2	
BREQkBranch if EqualIf $(Z = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRNEkBranch if Not Equalif $(Z = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRCSkBranch if Carry Setif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRCCkBranch if Carry Clearedif $(C = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRSHkBranch if Same or Higherif $(C = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRLOkBranch if Lowerif $(C = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRMIkBranch if Huseif $(N = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRPLkBranch if Greater or Equal, Signedif $(N = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRGEkBranch if Greater or Equal, Signedif $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRHSkBranch if Half Carry Flag Setif $(H = 1)$ then $PC \leftarrow PC + k + 1$ None1/2BRHCkBranch if Half Carry Flag Clearedif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if Telar Setif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2BRTSkBranch if Telar Setif $(H = 0)$ then $PC \leftarrow PC + k + 1$ None1/2	BRBC	S, K	Branch if Status Flag Cleared	if $(SREG(s) = 0)$ then $PC \leftarrow PC + k + 1$	None	1/2	
BRNEkBranch if Not EqualIf (Z = 0) then PC \leftarrow PC + k + 1None1/2BRCSkBranch if Carry Setif (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRPLkBranch if Minusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Telar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if Telar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2	BREQ	ĸ	Branch if Equal	if (Z = 1) then PC \leftarrow PC + k + 1	None	1/2	
BRCSkBranch if Carry SetIf (C = 1) then PC \leftarrow PC + k + 1None1/2BRCCkBranch if Carry Clearedif (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Telar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if Telar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2	BRNE	K	Branch if Not Equal	if $(2 = 0)$ then PC \leftarrow PC + k + 1	None	1/2	
brockkBranch if Carry ClearedIf (C = 0) then PC \leftarrow PC + k + 1None1/2BRSHkBranch if Same or Higherif (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Telar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if Telar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2	BRCS	ĸ	Didnoff II Carry Set	$ii (0 = 1) iiiei PO \leftarrow PO + K + 1$	None	1/2	
bronkBranch if Same of HigherIf (C = 0) then PC \leftarrow PC + k + 1None1/2BRLOkBranch if Lowerif (C = 1) then PC \leftarrow PC + k + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + k + 1None1/2BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if TElar Setif (H = 0) then PC \leftarrow PC + k + 1None1/2	BRUU	ĸ	Branch il Carry Cleared	II ($U = U$) then PC \leftarrow PC + K + 1	None	1/2	
BRUKBranch if MinusIf (U = 1) then PC \leftarrow PC + K + 1None1/2BRMIkBranch if Minusif (N = 1) then PC \leftarrow PC + K + 1None1/2BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + k + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half Carry Flag Clearedif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T Elar Setif (T = 1) then PC \leftarrow PC + k + 1None1/2	BRON DROM	ĸ		if (C = 1) then PC \leftarrow PC + K + 1 if (C = 1) then PC \leftarrow PC + k + 1	None	1/2	
DrivitKDraticit in WillingsIf (N = 1) then PC \leftarrow PC + K + 1None1/2BRPLkBranch if Plusif (N = 0) then PC \leftarrow PC + K + 1None1/2BRGEkBranch if Greater or Equal, Signedif (N \oplus V = 0) then PC \leftarrow PC + k + 1None1/2BRLTkBranch if Less Than Zero, Signedif (N \oplus V = 1) then PC \leftarrow PC + k + 1None1/2BRHSkBranch if Half Carry Flag Setif (H = 1) then PC \leftarrow PC + k + 1None1/2BRHCkBranch if Half Carry Flag Clearedif (H = 0) then PC \leftarrow PC + k + 1None1/2BRTSkBranch if T Elar Setif (T = 1) then PC \leftarrow PC + k + 1None1/2	BRLU	ĸ	Didnori ii Lower	if $(U = 1)$ then $PC \leftarrow PC + K + 1$	None	1/2	
BRGE k Branch if Greater or Equal, Signed If (N = 0) then PC \leftarrow PC + k + 1 None 1/2 BRGE k Branch if Greater or Equal, Signed if (N \oplus V = 0) then PC \leftarrow PC + k + 1 None 1/2 BRLT k Branch if Less Than Zero, Signed if (N \oplus V = 1) then PC \leftarrow PC + k + 1 None 1/2 BRHS k Branch if Half Carry Flag Set if (H = 1) then PC \leftarrow PC + k + 1 None 1/2 BRHC k Branch if Half Carry Flag Cleared if (H = 0) then PC \leftarrow PC + k + 1 None 1/2 BRTS k Branch if T Elar Set if (T = 1) then PC \leftarrow PC + k + 1 None 1/2	BRIMI	ĸ	Dranch ii Minus	II (IN = 1) then $PC \leftarrow PC + K + 1$	None	1/2	
Brace K Branch if Less Than Zero, Signed If (N \oplus V = 0) then PC \leftarrow PC + K + 1 None 1/2 BRIT k Branch if Less Than Zero, Signed if (N \oplus V = 1) then PC \leftarrow PC + k + 1 None 1/2 BRHS k Branch if Half Carry Flag Set if (H = 1) then PC \leftarrow PC + k + 1 None 1/2 BRHC k Branch if Half Carry Flag Cleared if (H = 0) then PC \leftarrow PC + k + 1 None 1/2 BRTS k Branch if T Elar Set if (T = 1) then PC \leftarrow PC + k + 1 None 1/2	BRPL	ĸ	Didition in Mus	if $(N = 0)$ then PC \leftarrow PC + K + 1 if $(N = 0)$ then PC \leftarrow PC + k + 1	None	1/2	
DRL1 K Diraticitii Less friant Zero, signed If (N \oplus V = 1) then PC \leftarrow PC + K + 1 None 1/2 BRHS k Branch if Half Carry Flag Set if (H = 1) then PC \leftarrow PC + K + 1 None 1/2 BRHC k Branch if Half Carry Flag Cleared if (H = 0) then PC \leftarrow PC + K + 1 None 1/2 BRTS k Branch if T Elar Set if (T = 1) then PC \leftarrow PC + K + 1 None 1/2		ĸ	Dranch in Greater or Equal, Signed	If $(N \oplus V = 0)$ then PC \leftarrow PC + K + 1 if $(N \oplus V = 1)$ then PC \leftarrow PC + k + 1	None	1/2	
Drino K Dranch if Half Carry Flag Set If (H = 1) then $PC \leftarrow PC + K + 1$ None 1/2 BRHC k Branch if Half Carry Flag Cleared if (H = 0) then $PC \leftarrow PC + k + 1$ None 1/2 BRTS k Branch if T Elag Set if (T = 1) then $PC \leftarrow PC + k + 1$ None 1/2		ĸ	Dranch if Helf Corry Flog Set	If $(N \oplus V = 1)$ then PC \leftarrow PC + K + 1 if $(H = 1)$ then PC \leftarrow PC + k + 1	None	1/2	
None Image: I	BBHC	k		if $(H = 0)$ then $PC \leftarrow PC + K + 1$	None	1/2	
	BRTS	k	Branch if T Flag Set	if $(T = 1)$ then PC \leftarrow PC + k + 1	None	1/2	

